EXHIBIT 1

FILED UNDER SEAL

CLEMENT SETH ROBERTS (STATE BAR NO. 209203)
croberts@orrick.com
BAS DE BLANK (STATE BAR NO. 191487)
basdeblank@orrick.com
ALYSSA CARIDIS (STATE BAR NO. 260103)
acaridis@orrick.com
EVAN D. BREWER (STATE BAR NO. 304411)
ebrewer@orrick.com
ORRICK, HERRINGTON & SUTCLIFFE LLP
The Orrick Building
405 Howard Street
San Francisco, CA 94105-2669
Telephone: +1 415 773 5700
Facsimile: +1 415 773 5759

SEAN M. SULLIVAN (*pro hac vice*)
sullivan@ls3ip.com
COLE RICHTER (*pro hac vice*)
richter@ls3ip.com
LEE SULLIVAN SHEA & SMITH LLP
656 W Randolph St., Floor 5W
Chicago, IL 60661
Telephone: +1 312 754 0002
Facsimile: +1 312 754 0003

Attorneys for Defendant Sonos, Inc.

UNITED STATES DISTRICT COURT

NORTHERN DISTRICT OF CALIFORNIA

SAN FRANCISCO DIVISION

GOOGLE LLC,

Plaintiff and Counter-defendant,

v.

SONOS, INC.,

Defendant and Counter-claimant.

Case No. 3:20-cv-06754-WHA
Related to Case No. 3:21-cv-07559-WHA

DECLARATION OF DR. DOUGLAS C. SCHMIDT IN SUPPORT OF SONOS, INC.'S OPPOSITION TO GOOGLE LLC'S MOTION FOR SUMMARY JUDGMENT PURSUANT TO THE COURT'S PATENT SHOWDOWN PROCEDURE

Date: June 9, 2022
Time: 8:00 a.m.
Place: Courtroom 12, 19th Floor
Judge: Hon. William Alsup

FILED UNDER SEAL

I, Douglas C. Schmidt, hereby declare as follows:

I. INTRODUCTION

1. I am the Cornelius Vanderbilt Professor of Engineering in the Department of Electrical Engineering and Computer Science at Vanderbilt University in Nashville, TN, where I also serve as the Associate Provost for Research Development and Technologies and the co-Director of the Data Science Institute. My research spans a broad range of software systems, including distributed object computing, middleware platforms, real-time operating systems, and distributed real-time and embedded systems. I became a Full Professor with tenure in 2003.

2. I received my Ph.D. degree in Computer Science from the University of California (UC) Irvine in Irvine, CA in 1994. I also earned a Master's Degree in Computer Science from UC Irvine in 1990. I first started programming in 1983 when I was an undergraduate student taking statistics courses. From 1985 through 1994 I learned how to program in Pascal, C, C++, Ada, Prolog, and Lisp, both at the College of William and Mary and at UC Irvine.

3. I have been a full-time university professor since 1994. I was previously a tenured professor at UC Irvine in the Electrical and Computer Engineering department, from 2000 to 2003, and before that at Wash. University in St. Louis, MO in the Computer Science and Engineering department and the Mallinckrodt Institute of Radiology, from 1994 to 1999. In addition, I served as the Chief Technology Officer and Deputy Director for the Software Engineering Institute (SEI) at Carnegie Mellon University from 2010 to 2012, where I led the SEI's research, development, and operational efforts related to software engineering and cyber-security.

4. For the past three decades, my research has focused on distributed real-time and embedded (DRE) systems, which has yielded the ACE, Java ACE, TAO, and CIAO middleware frameworks. The millions of lines of object-oriented code in these frameworks provide layers of infrastructure and distribution middleware that simplify the development of concurrent and networked software apps and services.

5. Besides my academic and research experience, from 2010 to 2014, I served as a member of the United States Air Force Scientific Advisory Board (SAB), where I was the Vice Chair of the SAB's Cyber Situational Awareness study, which conducted a comprehensive review

of the U.S. Air Force's tactics, techniques, and procedures related to secure network-centric mission operations. I also served on the Advisory Board for the U.S. Naval Air Systems Command Future Airborne Capability Environment and was a co-lead of a task force on "Published Open Interfaces and Standards" for the U.S. Navy's Open Systems Architecture initiative.

6. For over 30 years, I have conducted and supervised research projects involving a wide range of software-related topics, including patterns, optimization techniques, and empirical analyses of communication protocol stacks, web servers, and object-oriented middleware frameworks for distributed real-time embedded systems and mobile-/web-based cloud computing applications. I have published 650+ scholarly articles and technical papers, and I am the co-author/editor of 10+ books or book-length manuscripts on topics including software architecture, network programming, object-oriented frameworks, distributed and real-time systems, open-source middleware platforms, and web-/mobile-based cloud computing applications.

7. On top of my research experience, I have decades of hands-on programming experience with a variety of different programming languages. Starting in 1991, I led the development of one of the first C++ object-oriented frameworks for concurrent and networked middleware and applications (ACE). Starting in 1996, I developed one of the first Java object-oriented frameworks for concurrent and networked middleware and applications (Java ACE).

8. Since 1990, I have taught more than 2,500 students in dozens of face-to-face courses on network programming to both undergraduate and graduate students. Since 2013, I have taught mobile cloud computing to more than 400,000 students in online courses, which have focused on technologies like mobile app programming with Android, Java, and JavaScript, as well as programming cloud computing platforms using various web services frameworks.

9. Together with my regular course offerings, over the past 30 years I have also taught 600+ short-courses and tutorials on many subjects, including: software design patterns, object-oriented and functional programming; systems programming and network programming for UNIX and Windows; multi-threading and synchronization; concurrent and parallel programming; and various courses on distributed systems, real-time and embedded systems, TCP/IP, web apps and services, compiler construction, algorithms, and data structures.

II. SCOPE OF ASSIGNMENT AND INFORMATION CONSIDERED

10. I have been asked by Sonos, Inc. ("Sonos") to provide my opinions in response to certain non-infringement and invalidity arguments and opinions set forth by Google LLC ("Google") in its motion for summary judgment and by Google's expert Dr. Bhattacharjee in his supporting declaration regarding claim 13 of U.S. Patent 9,967,615 (the "'615 Patent"). I submit this Declaration in support of Sonos's opposition to Google's motion for summary judgment that is scheduled to be filed on May 5, 2022. The hourly rate for my services related to this matter is $550 per hour. My compensation is in no way contingent on the outcome of this action.

11. The language of claim 13 of the '615 Patent is set forth below with brackets containing Google's labeling of the claim limitations:

[13.0] A tangible, non-transitory computer readable storage medium including instructions for execution by a processor, the instructions, when executed, cause a control device to implement a method comprising:

[13.1] causing a graphical interface to display a control interface including one or more transport controls to control playback by the control device;

[13.2] after connecting to a local area network via a network interface, identifying playback devices connected to the local area network;

[13.3] causing the graphical interface to display a selectable option for transferring playback from the control device;

[13.4] detecting a set of inputs to transfer playback from the control device to a particular playback device, wherein the set of inputs comprises: (i) a selection of the selectable option for transferring playback from the control device and (ii) a selection of the particular playback device from the identified playback devices connected to the local area network:

[13.5] after detecting the set of inputs to transfer playback from the control device to the particular playback device, causing playback to be transferred from the control device to the particular playback device, wherein transferring playback from the control device to the particular playback device comprises:

(a) causing one or more first cloud servers to add multimedia content to a local playback queue on the particular playback device, wherein adding the multimedia content to the local playback queue comprises the one or more first cloud servers adding, to the local playback queue, one or more resource locators corresponding to respective locations of the multimedia content at one or more second cloud servers of a streaming content service;

(b) causing playback at the control device to be stopped; and

(c) modifying the one or more transport controls of the control interface to control playback by the playback device; and

[13.6] causing the particular playback device to play back the multimedia content, wherein the particular playback device playing back the multimedia content comprises the particular playback device retrieving the multimedia content from one or more second cloud servers of a streaming content service and playing back the retrieved multimedia content.

12. I understand that Sonos has accused Google of infringing claim 13 of the '615 Patent by virtue of, for example, making, using, offering to sell, selling, and/or importing in/into the United States computer devices (e.g., smart phones, tablets, smart displays, etc.) provisioned with any of the YouTube, YouTube Kids, YouTube TV, or YouTube Music apps (which I will refer to collectively as the "YouTube apps") or the Google Play Music ("GPM") app.

13. With respect to infringement, I understand that Google has only disputed limitation 13.5(a). Thus, the infringement opinions in this Declaration are limited to my evaluation of that specific claim limitation.

14. In performing my infringement analysis, I started by reviewing (i) the '615 Patent and its file history, (ii) Google's summary judgment motion and Dr. Bhattacharjee's supporting declaration and the other materials cited therein, and (iii) my expert report on claim construction for certain terms found in the '615 Patent that I submitted on February 11, 2022 (Dkt. 185-8; herein my "Claim Construction Report"). I followed that by evaluating the structure and operation of the YouTube apps and the GPM app, which involved a review of Google's source code (both on Google's source code inspection machine and in printed form), internal and publicly available materials, certain of Google's discovery responses in the present case (e.g., Google's Responses to Sonos's Interrogatory Nos. 14-15), and deposition transcripts from this case. I have also used a Pixel device provisioned with the YouTube apps to Cast to a variety of Google devices and have overseen and directed testing of the YouTube apps within an exemplary Google system (discussed below) to gain a further understanding of the structure and operation of the YouTube apps. Moreover, I reviewed the parties' contentions related to infringement of claim 13 of the '615 Patent, including Sonos's Infringement Contentions relating to '615 claim 13 and Google's

Response to Sonos's Interrogatory No. 12, as it relates to claim 13. After I completed my evaluation of the YouTube apps and the GPM app, I then compared the respective functionality of the YouTube apps and the GPM app to '615 claim 13 to determine whether a computer device provisioned with any of the YouTube apps or the GPM app satisfies the limitations of '615 claim 13 that Google is challenging in its motion for summary judgment.

15. With respect to validity, I understand that Google has asserted that claim 13 of the '615 Patent is anticipated by the YouTube Remote ("YTR") software application (installed on a computer device, such as a mobile phone) or, alternatively, obvious based on the YTR application in view of (i) the general knowledge of a POSITA and/or (ii) U.S. Patent No. 9,490,998 (the "'998 Patent"). Thus, the validity opinions in this Declaration are limited to my evaluation of those specific invalidity contentions.

16. In performing my validity analysis, I started by reviewing (i) the '615 Patent and its file history, (ii) Google's summary judgment motion and Dr. Bhattacharjee's supporting declaration and the other materials cited therein, and (iii) my Claim Construction Report. I followed that by evaluating the alleged prior art references, which involved a review of publicly available materials, as well as internal Google materials. Moreover, I reviewed the parties' contentions related to validity of '615 claim 13, including Google's Invalidity Contentions relating to '615 claim 13 and Sonos's Response to Google's Interrogatory No. 3, as it relates to claim 13.

17. A more detailed identification of the materials that I reviewed while evaluating Google's non-infringement and invalidity positions and formulating my opinions regarding the same can be found in the paragraphs that follow, as well as in Exhibit 1.

III. SUMMARY OF OPINIONS

18. Based on my infringement analysis, it is my opinion that each computer device provisioned with any of the YouTube apps or the GPM app literally satisfies limitation 13.5(a) of the '615 Patent. It is also my opinion that, even if either or both of Google's proposed constructions for "*playback queue*" and/or "*resource locator*" was adopted, each computer device provisioned with any of the YouTube apps or the GPM app would still satisfy limitation 13.5(a) literally and under the Doctrine of Equivalents ("DoE").

19. Based on my invalidity analysis, it is my opinion that Google and its expert Dr. Bhattacharjee have failed to prove that '615 claim 13 is anticipated or rendered obvious by any of the references and/or "general knowledge" of a POSITA identified in Google's motion and Dr. Bhattacharjee's supporting declaration – either considered alone or in combination.

IV. LEGAL STANDARDS

A. Infringement

20. I understand that a party directly infringes a patent whenever the party makes, uses, offers to sell, or sells a patented invention in the United States, or imports a patented invention into the United States, without authorization to do so from the patent holder.

21. I understand that, as the patent holder, it is Sonos's burden to prove direct infringement of '615 claim 13 by a preponderance of evidence. To do so, Sonos must show that it is more likely than not that a computer device provisioned with one of the YouTube apps or GPM app meets each and every limitation of '615 claim 13 literally or under DoE after being properly construed. Under DoE, I understand that an aspect of an accused product is equivalent to a limitation of an asserted claim if a POSITA would find the difference between the aspect of the accused product and the limitation of the claim to be insubstantial. Differences are often considered to be insubstantial if they perform substantially the same function, in substantially the same way, to achieve substantially the same result.

22. As shown above, '615 claim 13 is directed to a "tangible, non-transitory computer readable storage medium" (i.e., data storage) having "instructions for execution by a processor, the instructions, when executed, cause a control device to implement [the claimed] method" that follows. I understand that an accused device will meet the functional limitations of an "apparatus" claim structured in this way as long as the accused device is installed with software that makes it ***capable*** of performing the claimed functions. In this respect, the accused device need not be connected to a network, placed into a system with other devices, or actually used in a manner that causes the device to perform the claimed functions to meet the functional limitations of the "apparatus" claim. Thus, it is my understanding that '615 claim 13 is met by a computer device with data storage that is installed with software that is executable by the device's processor and

provides the computer device the capability to perform the functional limitations of the claim.

B. Validity

23. I understand that a patent claim is presumed valid until proven invalid by clear and convincing evidence. I understand that the '615 Patent is governed by pre-AIA 35 U.S.C. §§ 102-103 since the '615 Patent's priority date is before March 16, 2013.

a. Invention and Priority Dates

24. I understand that a party asserting invalidity based on prior art has the burden of proving by clear and convincing evidence that the material being relied upon by the party qualifies as prior art to a patent at issue. In assessing whether material qualifies as prior art, I understand that there are two dates with respect to the patent at issue that are important. The first is the "invention date," which is normally the date on which the inventor first conceived of the invention claimed in the patent at issue (assuming that the inventor was diligent in reducing the invention to practice). The second is the "priority date" (or "effective filing date"), which is the filing date of the earliest patent application to which the patent at issue is entitled to claim priority.

b. Qualification as Prior Art

25. I understand that a reference qualifies as prior art to a patent at issue under 35 U.S.C. § 102(a) only if the reference (1) "was known or used by others" in the U.S. before the invention date of the patent at issue, (2) is a patent that issued in the U.S. or a foreign country before the invention date of the patent at issue, or (3) is a printed publication that published in the U.S. or a foreign country before the invention date of the patent at issue.

26. I understand that a reference qualifies as prior art to a patent at issue under 35 U.S.C. § 102(b) only if the reference (1) is a patent that issued in the U.S. or a foreign country more than one year before the priority date of the patent at issue, (2) is a printed publication that published more than one year before the priority date of the patent at issue, or (3) was "in public use or on sale" in the U.S. more than one year before the priority date of the patent at issue.

27. I understand that a reference qualifies as prior art to a patent at issue under 35 U.S.C. § 102(g) only if the reference was made by another inventor who (1) either (a) reduced the invention to practice in the U.S. before the invention date of the patent at issue or (b) conceived of

the invention in the U.S. before the invention date of the patent at issue and exercised reasonable diligence in reducing the invention to practice after the invention date of the patent at issue and (2) did not abandon, suppress, or conceal the invention.

c. Anticipation

28. I understand that a patent claim is "anticipated" and therefore invalid only if a single prior art reference (e.g., published document, publicly available product/system, etc.) disclosed or embodied, either expressly or inherently, each and every element recited in the claim. I further understand that, to be considered anticipatory, a written prior art reference must be enabling and describe the invention of the claim sufficiently to have placed it in possession of a person of ordinary skill in the art ("POSITA") of the invention.

d. Obviousness

29. I understand that a patent claim is "obvious" and thus invalid under 35 U.S.C. §103 only if the differences between the claimed subject matter and the prior art are such that the subject matter as a whole would have been obvious to a POSITA at the time of the invention.

30. I understand that when a prior art reference is listed on the face of a patent, the prior art reference is presumed to have been considered during prosecution of the patent. I also understand that when a prior art reference was considered during prosecution of the patent, the party asserting invalidity has an added burden of overcoming the deference that is due to a government agency, such as the United States Patent and Trademark Office ("USPTO"), presumed to have properly done its job in evaluating the prior art reference.

31. In making an obviousness determination, I understand that there are several factors to consider: (1) the scope and content of the prior art; (2) the level of ordinary skill in the art at the time of the invention; (3) the differences between the claimed invention and the prior art, (4) whether there is a motivation to combine prior art references, and (5) objective evidence of non-obviousness (which is often referred to as "secondary considerations"), such as any commercial success, praise, copying, failure by others, licenses, longstanding need, and unexpected results.

32. I understand that prior art used to show that a claimed invention is obvious must be "analogous art." Prior art is analogous art to the claimed invention if (1) it is from the same field

of endeavor as the claimed invention or (2) it is reasonably pertinent to the problem faced by the inventor.

33. I understand that a patent claim composed of several elements is not proved obvious merely by demonstrating that each of its elements was independently known in the prior art, whether those references were set forth in a single reference or in a combination of references. Instead, it must have been obvious to combine those elements in the same way as the claimed invention does.

34. I understand that a single reference may be the basis for finding that a claim is obvious. When the obviousness determination relies on the combination of two or more references, however, the patent challenger must show that there is a reason, suggestion, or motivation that would lead a POSITA to combine prior art references.

35. I understand that, when relying on a combination of prior art references, the party asserting invalidity must submit objective evidence as to how and why the prior art references would have been combined to produce the claimed invention. This analysis typically includes evidence that a POSITA would have had an apparent reason or motivation to combine or modify the prior art in a manner that yields the claimed invention (e.g., a teaching or suggestion in the prior art), as well as evidence that a POSITA would have had a reasonable expectation that the combination would have succeeded.

36. I also understand that obviousness concerns whether a POSITA not only "could" have made but "would" have been motivated to make a combination or modification of prior art to arrive at the claimed invention.

37. I understand that when prior art teaches away from the claimed invention, that prior art ordinarily cannot be used to render the claimed invention obvious. In this regard, I understand that prior art teaches away from the claimed invention when a POSITA would be discouraged from following the path leading to the invention because of the prior art.

38. I understand that when separate prior art references teach away from one another or present conflicting teachings, these separate prior art references ordinarily cannot be combined to render the claimed invention obvious. In this respect, I understand that separate prior art

references teach away from one another when their combination would negate the original intent of the prior art or would produce a seemingly inoperative device.

39. I understand that it is improper to rely on hindsight when assessing obviousness. Many true inventions might seem obvious with the benefit of hindsight. I understand that the obviousness inquiry must be conducted from the standpoint of a POSITA at the time the claimed invention was made. In this regard, I understand that defining a problem in terms of its solution reveals improper hindsight in the selection of the prior art relevant to obviousness. What is known today, and what is learned from the teachings and disclosures of the patent itself containing the claim under analysis, should not be considered. Nor should one use the patent claim as a roadmap to picking out elements of the prior art for combination.

40. I understand that various secondary considerations (sometimes referred to as objective indicia of non-obviousness) may support a determination of non-obviousness and that such secondary considerations must be considered as part of an obviousness analysis. I understand that secondary considerations of non-obviousness may include factors such as praise of the invention.

41. I understand that the evidence of any secondary consideration must have a "nexus" to the claimed invention for the secondary consideration of non-obviousness to be given substantial weight, which means there must be a sufficient connection between the evidence and the claimed invention.

42. I understand that evidence that the industry or those skilled in the art praised the claimed invention or a product that embodies the claimed invention weighs in favor of non-obviousness. I understand that one rationale for this principle is that industry participants, especially competitors, are unlikely to praise an obvious advance over the prior art.

V. LEVEL OF ORDINARY SKILL IN THE ART

43. In my Claim Construction Report, I concluded that a POSITA for purposes of the '615 Patent is a person having the equivalent of a four-year degree from an accredited institution (typically denoted as a B.S. degree) in computer science, computer engineering, electrical engineering, or an equivalent thereof, and approximately 2-4 years of professional experience in

the fields of networking and network-based systems or applications, such as consumer audio systems, or an equivalent level of skill, knowledge, and experience. In forming the opinions set forth herein, I applied this level of ordinary skill in the art, but, my opinions would remain the same even if the level of ordinary skill were slightly different.

VI. OVERVIEW OF THE '615 PATENT

44. The '615 Patent stems from a filing on December 30, 2011 and generally describes a "local playback system" comprising one or more "playback devices" (or "zone players") that connect to a local "data network" (e.g., a home Wi-Fi network) and are capable of playing back multimedia content, such as audio. *See, e.g.*, '615 Patent, 1:13-15, 1:66-2:9, 2:51-3:13, 3:28-31, 5:21-54, 10:64-66, 12:44-67, 16:1-8. In this respect, the '615 Patent discloses that a "playback device" has a "local playback queue" for multimedia that the "playback device" is to playback. *See, e.g.*, *id.*, 16:20-31, 16:53-57, 16:63-17:4. The '615 Patent teaches that the "playback device" contains a resource locator (e.g., a URL, an identifier, or other reference) corresponding to a piece of multimedia content that facilitates the "playback device" accessing that multimedia content for playback, such as from the cloud. *See, e.g.*, *id.*, 11:62-12:3, 12:53-63, 13:31-40, 15:59-67. The '615 Patent also explains that a "playback device" can queue a single piece of multimedia content or multiple pieces of multimedia content for playback, which a POSITA would understand means that the "local playback queue" could contain a single resource locator corresponding to a piece of multimedia content or multiple resource locators corresponding to respective pieces of multimedia content. *See, e.g.*, *id.*, 9:27-31, 10:42-46, 11:65-12:3, 12:49-63, 13:33-40, 15:59-62, 16:63-17:4.

45. The '615 Patent further describes control devices (e.g., "network-enabled portable devices," such as smart phones) that also connect to the same local "data network" as the "playback devices" and are capable of controlling the operation of the "local playback system" (such a control device is sometimes referred to as a "controller"). *See, e.g.*, *id.*, 3:17-37, 4:53-5:28, FIG. 1.

46. Each "playback device" and control device can also communicate over a wide-area network, such as to retrieve audio from an Internet-based audio source. *See, e.g.*, *id.*, 5:38-41, 6:64-7:12, 12:44-67, FIG. 6. An illustrative example is shown in Figure 7 of a system architecture including a cloud-based "data network" (e.g., the Internet) and multiple "local playback systems"

on respective local "data networks" (760, 770). *See, e.g.*, *id.*, 12:19-43, 16:1-8, FIG. 7.

47. The '615 Patent discloses that control devices and "playback devices" may communicate with one another over a cloud-based "data network" to facilitate transferring playback from one device to another. For instance, the '615 Patent discloses a variety of situations where a user is listening on his/her personal computing device to music from an Internet-based, music application (e.g., Pandora, Rhapsody, Spotify, etc.) and decides to instead have that playback be transferred to one or more "playback devices" in his/her "local playback system." *See, e.g.*, *id.*, 12:44-13:30. The example system architecture shown in Figure 7 enables the user's personal computing device to communicate with one or more cloud-based servers to facilitate the transfer of playback from the personal computing device to one or more "playback devices" in a "local playback system." *See, e.g.*, *id.*, 12:19-43, 15:18-46, 16:1-8, 17:12-20. The '615 Patent further informs a POSITA that, in some embodiments, a "remote playback queue" may be involved in such a transfer of playback. *See, e.g.*, *id.*, 13:1-22, 16:63-17:4, 17:12-15, FIG. 7.

VII. INFRINGEMENT ANALYSIS

A. Brief Overview of the Accused Google Instrumentalities

48. As noted above, Sonos has accused Google of infringing '615 claim 13 in connection with computer devices (e.g., smart phones, tablets, smart displays, etc.) provisioned with (i) any of the YouTube apps or (ii) the Google Play Music (GPM) app. Google generally refers to such a device as a "Sender Device" (or just "Sender"). Mo Dep. Tr., 29:4-10; Patil Dep. Tr., 30:19-31:8. I may refer to a Sender installed with a YouTube app as a "YouTube Sender" and a Sender installed with the GPM app as a "GPM Sender." A Sender is provisioned with Google's "Cast" (or "Chromecast") technology that enables the Sender to transfer media playback responsibility from itself to a Cast-enabled media player, such as Google's Chromecast, Chromecast Audio, Chromecast Ultra, Chromecast with Google TV, Home Mini, Home, Home Max, Nest Mini, Nest Audio, Home/Nest Hub, Nest Hub Max, or Nest Wifi Point media player. *Id.*; Mo Dep. Tr., 29:4-31:1. Google generally refers to such a media player as a "Receiver Device" (or just "Receiver") or a "Cast Device." *Id.*

49. In general, a user may be experiencing multimedia playback (e.g., playback of

videos or songs) at a Sender device via one of the YouTube apps or the GPM app. The user can provide inputs at the Sender that initiate a Cast session with a selected Receiver that is on the same Wi-Fi network as the Sender. Patil Dep. Tr., 49:15-22. The Sender stops its own playback and typically causes one or more cloud servers to load resource locators for media items (e.g., videos or songs)[1] into a playback queue on the Receiver. The Receiver launches a "receiver app" corresponding to the app that the Sender was running when the user initiated the Cast session and uses the resource locators to retrieve the media items from one or more cloud servers for playback. *See, e.g.*, *id.*, 59:5-18, 74:22-75:15; Mo Dep. Tr., 82:3-18, 105:1-109:25.[2] In this regard, playback of media is transferred from the Sender to the Receiver. Later, I discuss more specific details regarding Casting from one of the YouTube apps versus Casting from the GPM app.

B. Testing of the YouTube Apps

50. As noted above, part of my infringement analysis involved overseeing and directing certain testing of the YouTube apps within an exemplary Google system to further understand the structure and operation of the YouTube apps. In particular, the Google system, which I refer to as the "YouTube Test System," included the following devices that were all communicatively coupled via a Wi-Fi network: (i) a Pixel 6 running Android version 12 provisioned with the YouTube app (version 17.14.35), the YouTube Music app (version 5.02.50), the YouTube Kids app (version 7.12.1), and the YouTube TV app (version 6.14.0); (ii) a Nest Hub Max player running Cast firmware version 1.56.295071, which was named "Nest Hub Max"; and (iii) a Nest

[1] In the context of the '615 Patent, the phrase "multimedia item" is synonymous with "media item," so I may use these phrases interchangeably. *See, e.g.*, '615 Patent, 1:19-21 ("Technological advancements have increased the accessibility of music content, as well as other types of media, such as television content, movies, and interactive content."), 2:6-8 ("Music and/or other multimedia content can be shared among devices and/or groups of devices …."), 12:61-63 ("Songs and/or other multimedia content can be retrieved from the Internet ….").

[2] *See also, e.g.*, GOOG-SONOSWDTX-00006780, 83 ("receiver app[:] [t]he receiver app receives commands from the sender app …. For example, the YouTube app on Chromecast. See receiver."), ("receiver[:] [a] receiver is an application created using HTML, JavaScript, and CCS. It is loaded onto a Cast device (for example, a Chromecast) through a URL that is accessible over the Wi-Fi network …."), ("Web Receiver[:] [a] Web Receiver application is an HTML5/JavaScript application that runs on the receiver device, such as Chromecast."); GOOG-SONOSWDTX-00039916, 22 (illustrating different HTML5 receiver apps for YouTube apps).

Audio player running Cast firmware version 1.54.279716, which was named "Nest Audio 1."

51. Unless I specify otherwise below, the screenshots included herein were captured by the Pixel 6 while operating within the YouTube Test System. I reserve my right to conduct a demonstration of the functionality of the YouTube Test System and/or to present additional screenshots illustrating the use and testing of the YouTube Test System.

C. Response to Dr. Bhattacharjee's Non-Infringement Analysis

52. The following section sets forth a detailed discussion of my analysis of Dr. Bhattacharjee's non-infringement opinions with regard to '615 claim 13, my own infringement analysis, and the opinions I have reached based on that analysis.

53. As explained in detail below, it is my opinion that each computer device provisioned with any of the YouTube apps or the GPM app (i) literally satisfies limitation 13.5(a) of the '615 Patent, which is the only limitation that Dr. Bhattacharjee contested, and (ii) satisfies limitation 13.5(a) literally and under DoE even if either or both of Google's proposed constructions for "playback queue" and/or "resource locator" was adopted.

1. Casting Involves a "Local Playback Queue" on the Receiver

54. Limitation 13.5(a) of the '615 Patent recites "*causing one or more first cloud servers to add multimedia content to a local playback queue on the particular playback device* ...."[3] I have seen considerable evidence where Google acknowledges that a Receiver has a local queue that dictates its playback when in a Cast session.

55. For example, Google's source code includes comments indicating that Google's engineers view a Receiver as maintaining a local queue. *See, e.g.*, CloudQueueSyncCoordinator.java[4], lns. 35-36 ("Notifies the receiver about any local changes to the cloud queue and asks the ***receiver***[5] to refresh ***its queue***."); application.ts[6], lns. 527-29 ("videoQueuer" "[m]anages the ***video queue*** and transitions."), lns. 3519-22 ("isQueueEmpty"

[3] For readability and context, I have italicized claim language throughout this Declaration.
[4] /2020-09-01-google_play_music_android_8.27.8839-3.U/java/com/google/android/libraries/play/music/playback2/players/
[5] Emphasis has been added herein unless I specify otherwise.
[6] /2021-02-01_YTReceivers09292020/google3/video/youtube/web/player/

"True if there are no videos in the ***queue***"). Google's engineers also describe a Receiver as having a queue in internal correspondences. *See, e.g.*, GOOG-SONOSNDCA-00072008, 2009 ("[R]eceivers can ***retain a local version of the remote queue***…. It sounds like the ***queue is intentionally retained*** to guard against flaky networks …."). Consistent with Google's internal recognition, Google publicly describes a Receiver as maintaining a playback queue. *See, e.g.*, GOOG-SONOSWDTX-00006865, 66 ("The ***Receiver SDK maintains the queue*** and responds to operations on the queue as long as the queue has at least one item currently active (playing or paused)."); GOOG-SONOSWDTX-00006780, 82 ("Cast utilizes both a basic sender-initiated queue and ***receiver-implemented queueing***.").

56. As noted in some of the above commentary by Google, the Receiver's local queue is kept in sync with a "remote queue" stored in the cloud (or "cloud queue") by virtue of the Receiver retrieving or loading segments of the remote queue into its local queue. *See also, e.g.*, GOOG-SONOSWDTX-00039480, 82 (showing "Receiver" will "***[r]etrieve*** remote queue" at start of a Cast session); GOOG-SONOSWDTX-00043467, 67 ("Tells the receiver to ***load*** the cloud queue."); GOOG-SONOSWDTX-00006865, 66 ("The ***receiver maintains a session for queue items*** until the last item completes playback … or until a sender ***loads a new queue on the receiver***."); GOOG-SONOSWDTX-00006873, 74 (explaining that Cast's "[q]ueuing" functionality provides "[s]upport of Google's … cloud queue implementation so externally stored and created ***queue*** can be ***directly loaded into Cast devices***.").[7]

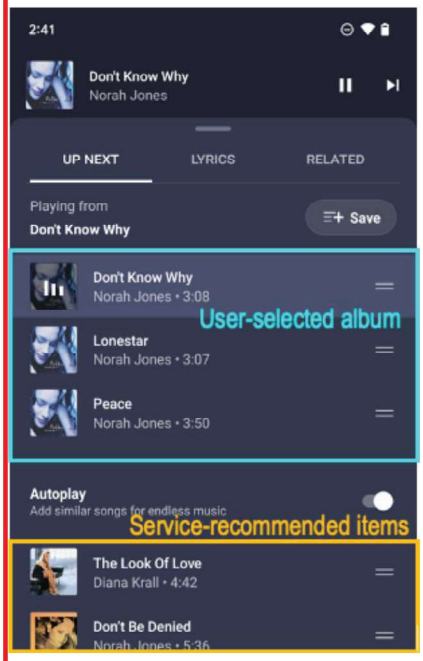
57. Similarly, the Sender has a local queue that syncs with the remote queue when in a Cast session with a Receiver. *See, e.g.*, GOOG-SONOSWDTX-00039916, 43 ("[YouTube Music Sender's] MDx integration includes a ***local queue*** that is ***kept in sync*** with the ***remote queue***."). In this way, the Receiver and Sender "share a queue state" when in a Cast session. GOOG-SONOSWDTX-00051490, 90 ("While Casting, YTM [Senders] and Cast receivers ***share a queue state*** via MDx's 'remote queue'.").

[7] Much like a Receiver in a Cast session, Google describes a Sender/client "loading" its local queue with "queue elements" from a queue maintained by a cloud service prior to Casting. *See, e.g.*, GOOG-SONOSWDTX-00039798, 98-99; GOOG-SONOSWDTX-00039785, 88-89.

58. As I explain in detail below, Google implemented the Receiver's local queue while in a YouTube or Google Cast session in different manners. Regardless, it is my opinion that both implementations infringe limitation 13.5(a).

a. Casting YouTube Involves a "Local Playback Queue"

59. A YouTube Sender can operate in a non-Casting state in which the Sender itself plays back media or a Casting state in which the Receiver plays back media after a Cast session is initiated via the Sender. Mo Dep. Tr., 130:6-17. While in the non-Casting state, each YouTube app running on a Sender enables a user to select a single media item (e.g., a song or video) for playback at the Sender, and each YouTube app except for the YouTube TV app allows a user to select a collection of media items (e.g., an album or playlist) for playback at the Sender. Based on the initial user-selected media item or collection of media items, a Google cloud service will automatically recommend one or more additional media items that should be played by the Sender after the initial media item or collection (for some of the YouTube apps, an "Autoplay" setting must be enabled for this to occur). *See, e.g.*, GOOG-SONOSWDTX-00052121, 27-28 ("'Watch next' functionality used to be called 'Related videos' and now is called 'Up next,' and has an Autoplay toggle…. Discovery technology compiles recommendations …."). As a result, the Sender's local queue is loaded with one or more "videoIds[8]" for the initial user-selected media item or collection of media items and, at least in some cases, one or more videoIds for the service-recommended media item(s) seeded by the initial user selection. *See, e.g.*, GOOG-SONOSWDTX-00039798, 99 ("The GetMusicWatchNext endpoint provides queue renderers when initiating playback on a new container, on queue continuation loads, and for the autoplay section of the queue."); Nicholson Dep. Tr., 101:3-9 ("[T]here's a video ID in those renderers."). A representation of the Sender's



[8] YouTube uses a videoId regardless of whether the multimedia item is a song or video.

local queue is typically displayed by the Sender, with one example shown here for YouTube Music. *See also, e.g.*, GOOG-SONOSWDTX-00039798, 98.

60. To transition from the non-Casting state to the Casting state and thereby transfer playback from the Sender to a Receiver, the user selects a "Cast icon" button displayed by the YouTube Sender and then selects a particular Receiver that the user would like to experience media playback on. An example illustration of these user selections is shown in Dr. Bhattacharjee's Declaration at paragraph 43. While in the Casting state, the Sender stops its own playback and can be used to control the selected Receiver's media playback. *See* Mo Dep. Tr., 133:22-134:5.

61. After receiving the user selections, the Sender transmits a "setPlaylist" message to an "MDx session server" in the cloud that in turn sends a setPlaylist message to the Receiver. *See, e.g.*, *id.*, 112:21-115:14, 116:12-22, 117:17-118:14, 129:25-130:5, 135:19-136:2. The setPlaylist message typically includes, among other information, a videoId corresponding to the multimedia item (e.g., song or video) that the Receiver is to begin playing, which is the item that was to be played by the Sender when it received the user selections to Cast. *See, e.g.*, *id.*

62. In response to receiving a setPlaylist message with a videoId, the Receiver loads the videoId in its queue, thereby updating its internal logic such that the videoId corresponds to the media item the Receiver is set to currently playback. *See, e.g.*, *id.*, 136:3-21, 151:10-17 (Google 30(b)(6) witness admitting videoId of current video stored in Receiver's memory "is what's used in subsequent API calls that eventually results in the get play[er] request."); remote.ts → getCurrentVideoId, lns. 1369-76 ("currentWatchEndPoint should be the source of truth ...."). While Google's 30(b)(6) designee could not definitively identify the name of the data variable where the current videoId is stored that drives the Receiver's playback of the corresponding media item (Mo Dep. Tr., 150:15-151:17, 152:11-153:7, 154:14-155:10), I understand that Google contends that the relevant data variable is PlaybackParams.videoId assigned in the handleMessage function. *See* loungeadapter.ts,[9] lns. 940-52.

63. By virtue of the Sender transmitting the setPlaylist message to the MDx server, the

[9] 2021-02-01_YTReceivers09292020/google3/video/youtube/src/web/javascript/library/mdx/screen_ts/

Sender also causes a "WatchNext" server to transmit a "WatchNextResponse" (WNR) to the Receiver. *See, e.g.*, Mo Dep. Tr., 155:21-156:22, 157:16-158:7; 169:20-170:4, 170:18-171:22. The Receiver stores the WNR in memory and can use various information contained in the WNR for playback. *See, e.g.*, *id.*, 166:19-167:9, 168:8-21 ("I'm sure there is some variable somewhere that represents the watch next response…. I would imagine that there is a variable somewhere that stores the autoplay information."), 175:1-5 (in response to question "[w]ill the cast receiver store the video ID of the next video somewhere in memory," admitting,"[i]n the case where the watch next response contains that, yes, because I would imagine the watch next response is – the model is stored in memory."), 178:17-179:10; Nicholson Dep. Tr., 113:25-115:7, 115:11-18, 140:24-141:9.

64. The contents of the WNR may vary depending on the state of the YouTube Sender's playback when the user initiated the Cast session, but the WNR typically will include at least a videoId for the media item that the Receiver is set to currently play back, as well as a videoId for the next media item that the Receiver is set to play back, among other possible videoIds.[10] *See, e.g.*, Mo. Dep. Tr., 170:23-172:24 (explaining WNR "may contain the current watch video endpoint" and "could contain [fields] with video IDs that may differ … from the current. There could be – there is a notion of – there's autoplay set renderer where it will contain like next, previous, so forth, depending on the playback."), 174:10-175:23, 178:17-179:10; GOOG-SONOSWDTX-00039491, 91 ("When [Receiver] plays a video in the queue, the Watch Next service will send the video id … for the next video in the queue so that [Receiver] knows what to play when the current video finishes."); innertube_service.proto,[11] lns. 822-32 ("Returns videos to play_next (related + autoplay + radio + playlist)."); get_next_playlist_video.ts,[12] lns. 9-24 (referencing "nextVideoRenderer" in connection with "[e]ndpoint of the next video in the

[10] I understand Google's 30(b)(6) witness was unable to identify the specific name of the data variable that stores the next videoId. Mo. Dep. Tr., 163:17-164:21, 171:23-172:12, 175:1-16.
[11] /2021-02-02_YTServerInnerTubeWatchNext09292020/google3/video/youtube/api/innertube/proto/
[12] /2021-02-01_YTReceivers09292020/google3/video/youtube/tv/bedrock/ts/watch/commands/

playlist."); remote.ts,[13] lns. 1721-54 ("handleWatchNextLoaded" function defining "upNextVideoId" variable).

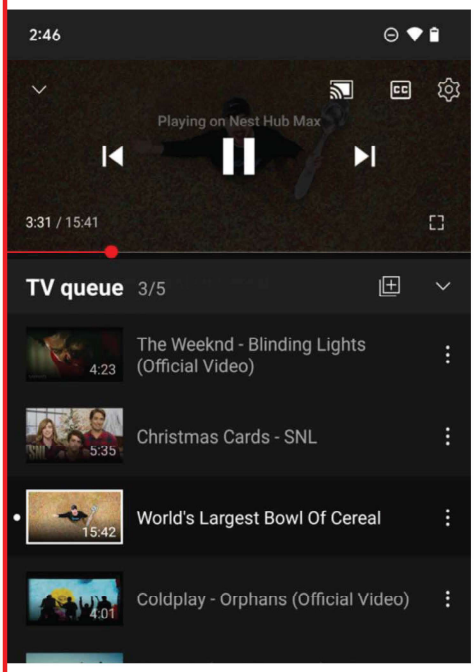
65. More specifically, a WNR includes an "AutoplayRenderer" which in turn typically includes several "AutoplaySets."[14] *See, e.g.*, GOOG-SONOSWDTX-00051490, 90 ("Cast receiver devices determine the next track to play through the [WNR]'s AutoplayRenderer…. The renderer includes a list of AutoplaySets for the clients to select from …. The AutoplaySets each contain a renderer for the next video to play for the given mode."); Nicholson Dep. Tr., 120:9-121:5 ("So the Autoplay set is included within the Autoplay renderer and there may be more than one Autoplay set in that list included in the Autoplay renderer."), 121:24-122:21. Each AutoplaySet corresponds to a "playback scenario" that the Receiver might perform and contains a videoId used when the Receiver is to perform the given playback scenario. *See, e.g.*, *id.*, 120:9-121:5 ("And there's also the video ID somewhere within that Autoplay set. So depending on what the action is that the receiver is trying to take, it will select one of these Autoplay sets and take the video ID from there and then make a Get Player call with that video ID and so on and so on."), 124:2-4; GOOG-SONOSWDTX-00051490, 512 ("Within those renderers is a watch endpoint that should have params/video id …."); Mo Dep. Tr., 171:13-172:24 (Google's 30(b)(6) witness admitting WNR can contain "auto playset renderer where it will contain like next, previous, so forth, depending on the playback" which serves as "information for the client to handle different playback scenarios" and "the endpoints in the autoplay set renderers could contain different video ID values.").

66. I will provide an example to help describe what this means in practice. In the screenshot here, a user had queued a playlist of five videos for playback at a YouTube Sender and was watching the third video in that playlist when they initiated a Cast session on a Receiver. The

[13] /2021-02-01_YTReceivers09292020/google3/video/youtube/tv/bedrock/ts/mdx/

[14] "AutoplayRenderer" and "Autoplayset" are different from the user-facing "Autoplay" feature that provides service-recommended media items for playback, although an "Autoplayset" can include a videoId of a service-recommended media item. Mo Dep. Tr., 171:23-172:8.

Receiver received a WNR that included at least an Autoplayset for the current video being played (i.e., "World's Largest Bowl Of Cereal"), an Autoplayset for the previous video (i.e., "Christmas Cards - SNL"), and an Autoplayset for the next video (i.e., "Coldplay - Orphans"). *See, e.g.*, autoplay_renderer.proto,[15] ln. 71 (next_video_renderer), ln. 75 (previous_video_renderer); watch_next_response.ts,[16] ln. 21 (currentVideoEndpoint). In turn, each Autoplaylist contained a videoId, and the Receiver stored the WNR such that it had the videoId for the current video, the previous video, and the next video.



67. Now, if the user tapped the skip backwards transport control icon on the Sender, the Receiver would retrieve from memory the previous videoId to use to retrieve the "Christmas Cards - SNL" video for playback. *See, e.g.*, Nicholson Dep. Tr., 132:9-134:9 ("There is an Autoplay set that contains the video ID for what should be the previous video in this list and there will be some mode and that mode will be titled something that does suggest the previous one …."). Similarly, if the user tapped the skip forward transport control icon on the Sender or just allowed the Receiver to complete playback of the "World's Largest Bowl Of Cereal" video, the Receiver would retrieve from memory the next videoId to use to retrieve the "Coldplay - Orphans" video for playback. *See, e.g.*, *id.*, 132:9-134:9 ("[I]f you're in the casting scenario … and you hit the next track button, the receiver device should use the Autoplay set that has the video ID for the next video ID in that remote queue playlist …."), 139:22-140:22.

68. As suggested before, the Receiver uses a videoId to locate the corresponding media item. *See, e.g.*, Mo. Dep. Tr., 137:17-25, 138:13-23, 139:13-140:9, 141:8-15, 142:6-144:15, 146:1-7. In particular, the Receiver makes a "getPlayer" call to a "Player Service" cloud server

---

[15] /2021-02-02_YTServerInnerTubeWatchNext09292020/google3/video/youtube/api/innertube/proto/watch_next/renderers/

[16] /2021-02-01_YTReceivers09292020/google3/video/youtube/tv/bedrock/ts/innertube/models/

with the videoId and receives a response containing at least one URL for the media item. *See, e.g.*, *id.*; GOOG-SONOSWDTX-00039491, 91; GOOG-SONOSWDTX-00039785, 85-86. The Receiver then uses the returned URL to retrieve the media item for playback from one or more "Bandaid" cloud servers that differ from the MDx and WatchNext servers. *See, e.g.*, *id.*; Mo. Dep. Tr., 142:6-144:15.

69. Returning to '615 claim 13, limitation 13.5(a) recites "*causing one or more first cloud servers to add multimedia content to a local playback queue on the particular playback device* ...." In evaluating whether this element is satisfied, I applied the plain and ordinary meaning to the term "*local playback queue*" in the context of the '615 Patent: a data construct[17] on the playback device that can contain one or more resource locators (e.g., videoIds, URLs, or other resource locators), where each resource locator corresponds to multimedia content (e.g., a particular song or video) that the playback device is to playback. *See* Dkt. 185-8, ¶59.

70. It is my opinion that the "*local playback queue*" element is satisfied because, when a YouTube Sender Casts to a Receiver, the Sender is configured to cause the MDx session server to add a videoId to the Receiver's PlaybackParams.videoId data variable that dictates what media item the Receiver is to currently playback. *Supra* ¶¶61-62. This behavior is evident from the getCurrentVideoId function referencing data variables currentWatchEndPoint.videoId and currentVideoIdDeprecated that are set to this same videoId value. *See* remote.ts,[18] lns. 1370-76.

71. It is also my opinion that the "*local playback queue*" element is satisfied because, when a YouTube Sender Casts to a Receiver, the Sender is configured to cause the WatchNext server to add to the Receiver's memory a WNR that often includes at least a respective videoId of the current and next media item that the Receiver is to playback (among possibly numerous other videoIds corresponding to other items for playback) and the next videoId is added to an upNextVideoId data variable, as well as a nextVideoRenderer data variable, each of which dictates what item the Receiver is to playback next after the current item. *Supra* ¶¶63-67. Thus, even if

[17] As I explained in my Claim Construction Report, a data construct can take the form of a single data variable, multiple data variables, a data array, or other data structure. *See* Dkt. 185-8, ¶58.

[18] /2021-02-01_YTReceivers09292020/google3/video/youtube/tv/bedrock/ts/mdx/

Dr. Bhattacharjee was correct (he is not) that a "*playback queue*" must "allow for the storage of multiple items" (Bhatta. Decl., ¶70), the combination of the variables storing the current and next videoIds satisfies this requirement, as does the WNR stored on the Receiver.

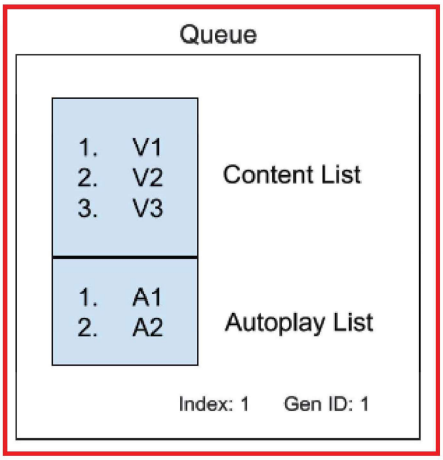
72. To avoid this conclusion, Dr. Bhattacharjee agrees with the construction that Google advanced for "*playback queue*" – an ordered list of multimedia items that is selected by the user for playback – and contends that YouTube fails to meet this construction. *See, e.g.*, Bhatta. Decl., ¶68. Yet, even under this erroneous construction, it is my opinion that the "*playback queue*" of limitation 13.5(a) is still satisfied.

73. As an initial matter, I set forth in detail in my Claim Construction Report why I disagree with Google's proposed construction for "*playback queue*" (Dkt. 185-8, ¶¶41-95), but it is notable that I have seen plenty of evidence that Google's proposed construction is contrary to how Google itself refers to the concept of a playback queue. For example, despite Google's construction requiring plural "multimedia items," one of its 30(b)(6) witnesses admitted that "one item [can] exist in a queue[.]" Patil Dep. Tr., 42:3-4. Likewise, another of Google's witnesses explained that "it's possible to have an empty queue" and that "the MDx queue can be empty" or "can have a single item or – or more." Levai Dep. Tr., 108:3-18. Similarly, comments in Google's YouTube Receiver-side source code demonstrates that Google's engineers believe a Receiver's queue can be limited to containing only a single media item. *See, e.g.*, video_queuer.ts,[19] lns. 17-22 ("Supports queueing videos. Currently this only supports a single video in the queue.").

74. As another example, despite Google's construction limiting a queue to items "selected by the user for playback," I have seen several internal Google documents describing a queue as containing one or more items automatically recommended by one of Google's services based on a user's selection of a single media item or collection of media items. *See, e.g.*, GOOG-SONOSNDCA-00073352, 56 ("Tap on a single video. An implicit list is created for the user, using autonav. We'll generate several look-ahead items to auto-extend the queue…. The look-ahead can contain videos which were explicitly queued-up by the user, and it contains videos automatically

[19] /2021-02-01_YTReceivers09292020/google3/video/youtube/web/player/playback/

added by autoplay."); GOOG-SONOSWDTX-00039798, 99 ("[T]he autoplay section of the queue."); GOOG-SONOSNDCA-00073394, 95 (illustration, reproduced here, of queue containing "Autoplay" section). Thus, it must be the case that Google does not mean for its construction to exclude automatically-recommended items seeded by an initial user selection.



75. In any event, it is my opinion that at least the Receiver's storage of a WNR literally satisfies the "ordered list of multimedia items" aspect of Google's proposed construction. Indeed, a WNR can specify, for example, the item that the Receiver is currently set to play, the item before the current item, and the item after the current item. In this way, the WNR is a data construct that can provide multiple videoIds and specify the order in which the Receiver is to playback the corresponding multimedia content.

76. In addition, it is my opinion that the combination of the data variables stored by the Receiver containing the videoIds of the current and next media items satisfies the "ordered list of multimedia items" aspect of Google's proposed construction under DoE. In this regard, it is my opinion that there is merely an insubstantial difference between a Receiver having (i) a singular data structure that provides an ordered list of the current and next videoIds for playback and (ii) a combination of data variables that collectively provides current and next videoIds that the Receiver knows to use for playback in a specific order. In fact, a POSITA would recognize the difference to be nothing more than an implementation detail of how to achieve a playback queue rather than a functional difference.

77. Indeed, regardless of whether a Receiver has a singular data structure or a set of data variables that provides videoIds for playback in a particular order, the Receiver performs substantially the same function (e.g., adding multimedia content to its "*local playback queue*"), in substantially the same way (e.g., by adding a current and next videoId to respective locations in the Receiver's memory), to achieve substantially the same result (e.g., maintaining a "*local playback queue*" that includes multiple videoIds that the Receiver knows to use for playback in a

specific order) as a Receiver having a singular data structure that provides an ordered list of videoIds of the current and next media items for playback. Moreover, regardless of whether a Receiver has a singular data structure or a set of data variables that provides videoIds for playback in a particular order, the Sender performs the same function (e.g., causing playback of the corresponding multimedia content to be transferred), in the same way (e.g., by transmitting a "setPlaylist" message to the MDx server), to achieve the same result (e.g., causing the MDx and WatchNext servers to add the current and next videoIds to the Receiver's memory such that Receiver knows the order in which it should use the videoIds for playback).

78. Returning to Google's proposed construction, it is also my opinion that (i) the current and next videoId data variables and (ii) the WNR both satisfy the "selected by the user for playback" aspect of Google's construction.

79. For example, as I explained before, each YouTube app allows a user to select an initial media item for playback, which results in a cloud service recommending at least one additional media item for playback. *Supra* ¶59. In such a scenario, the videoId that is added to the Receiver's "*local playback queue*" at the time of Casting is for the initial media item that was directly ***selected*** by the user for playback, and the next videoId that is added to the Receiver's "*local playback queue*" is for a service-recommended media item that was indirectly ***selected*** by the user as it was selected based on the user's direct selection of the initial media item. Thus, each YouTube Sender at least literally satisfies the "*local playback queue*" element under Google's construction.[20]

80. As another example, as I explained before, the YouTube, YouTube Music, and YouTube Kids apps allow a user to select a collection of media items for playback. *Supra* ¶59. In such a scenario, the videoIds that are added to the Receiver's "*local playback queue*" at the time of Casting (e.g., for at least the current and next media items in the collection) were each "selected by the user for playback." Thus, each Sender installed with any of the YouTube, YouTube Music,

[20] Dr. Bhattacharjee did not dispute that a service-recommended media item that is seeded based on an initial media item that was directly selected by the user amounts to being "selected by the user for playback" in Google's construction, which makes sense given Google's own words describing a queue in its system. *Supra* ¶¶73-74.

or YouTube Kids apps literally satisfies the "*local playback queue*" element under Google's construction for this additional reason.

81. For at least the above reasons, it is my opinion that YouTube satisfies the "*local playback queue*" element even under Google's proposed construction.

b. Casting GPM Involved a "Local Playback Queue"

82. GPM was discontinued, but I understand that a GPM Sender generally functioned in a similar manner as a YouTube Sender up to the point at which the GPM Sender transitioned from a non-Casting to a Casting state. *Supra* ¶¶59-60; SONOS-SVG2-00067554.

83. After receiving user selections at the GPM Sender to Cast to a Receiver, the Sender would transmit a "setQueue" message to a Cloud Queue (CQ) server to sync a cloud queue at the CQ server with the Sender's local queue. *See, e.g.*, Patil Dep. Tr., 76:12-77:11. As one example, if the user selected a collection of songs (e.g., a playlist or album) to play back at the Sender before Casting, the setQueue message would cause the CQ server to populate the cloud queue with that collection of songs. *See, e.g.*, *id.*, 36:22-37:10, 47:15-48:4, 76:22-77:11.

84. The Sender would then transmit a "Load" message causing the Receiver to load the current media item and contact the CQ server for a window of media items around the current media item from the cloud queue (via a "getItemWindow" request), which in turn caused the CQ server to send to the Receiver an "ItemWindowResponse" (IWR) containing a respective "itemID" and "trackUrl" for (i) the current media item, (ii) the media item before the current media item, and (iii) the media item after the current media item. *See, e.g.*, *id.*, 77:12-23, 81:22-82:4, 86:5-18; GOOG-SONOSWDTX-00043467, 69. The Receiver loaded the contents of the IWR into its queue (e.g., in an itemWindowResponse.items data structure), thereby updating its internal logic such that the Receiver knew what particular itemID/trackUrl pair corresponded to the current media item ("baseItem," "currentCloudQueueItem_"), the item before the current media item ("previousItem," "previousCloudQueueItem_"), and the item after the current media item ("nextItem," "upcomingCloudQueueItem_"). *See, e.g.*, Patil Dep. Tr., 86:19-88:9, 89:21-90:9,

91:24-92:5; playermanger.ts[21], lns. 1887-89 (defining "baseItem," "previousItem," "nextItem"), ln. 1916 (defining "upcomingCloudQueueItem_"), ln. 1919 (defining "previousCloudQueueItem_"), ln. 1953 (defining "currentCloudQueueItem_").

85. The Receiver would then use a given itemID/trackUrl pair to retrieve the media item for playback from one or more Bandaid cloud servers that differed from the CQ server. *See, e.g.*, GOOG-SONOSWDTX-00043467, 69; Patil Dep. Tr., 104:11-105:10, 109:21-110:7. For instance, if the user provided input at the Sender to play the media item before the currently playing media item (e.g., via a skip backwards transport control icon), the Receiver would retrieve from its memory the itemID/trackUrl pair corresponding to the "previous" media item and use that itemID/trackUrl pair to retrieve the corresponding music. *See, e.g.*, *id.*, 103:22-104:9 (in response to "[w]hen a user selected the previous track transport control, is the previous item ID and URL used, or is another item window request made," answering "[t]he receiver starts playing the previous track and while it is playing the previous track, it then makes an item window [request] to fetch the pointers that are – that correspond to a track above the currently played track now."). Similarly, if the user provided input at the Sender to play the next media item or just allowed the Receiver to complete playback of the current media item, the Receiver would retrieve from its memory the itemID/trackUrl pair corresponding to the next media item and use that itemID/trackUrl pair to retrieve the corresponding music. *See, e.g.*, *id.*, 100:18-102:3 (in response to "if a user selects the next song transport control, a get item window call would be made by the receiver; correct," answering "[t]he receiver app then starts playing the next track and then initiates the item window request to the Cloud queue.").

86. It is my opinion that GPM satisfied the "*local playback queue*" element under the plain and ordinary meaning of that term and under Google's construction because, when a GPM Sender Casted to a Receiver, the Sender was configured to cause the CQ server to add to the Receiver's memory an IWR containing a respective itemID/trackUrl pair for three multimedia items (i.e., songs) that were selected by the user for playback. Much like the WNR in the YouTube

[21] /2021-01-27_GPMReceiver09292020/google3/java/com/google/wireless/android/skyjam/frontend/javascript/cast/

scenario, an IWR in GPM specified the song the Receiver was currently set to play, the song before the current song, and the song after the current song. *Supra* ¶¶63-67, 71, 75. In this way, the IWR was a data construct that provided multiple itemID/trackUrl pairs and specified the order in which the Receiver was to playback the corresponding multimedia content, thereby amounting to a "*local playback queue*."

2. Dr. Bhattacharjee's "Queue" Arguments Fail

87. Dr. Bhattacharjee provides a host of arguments to support his opinion that neither Casting YouTube nor Casting GPM involves a "*local playback queue*," but each argument fails.

a. A "Local Playback Queue" & "Cloud Queue" Are Not Mutually Exclusive

88. For both YouTube and GPM, at the core of Dr. Bhattacharjee's opinions is the faulty premise that a "*local playback queue*" and a "Cloud Queue" are mutually exclusive. *See, e.g.*, Bhatta. Decl., ¶¶63-66, 83, 113-20. In other words, Dr. Bhattacharjee contends that a queue can only exist at one device in a multi-device system, such as Google's system and the system disclosed in the '615 Patent. I disagree for several reasons.

89. ***First***, I see nothing in claim 13 that precludes the possibility that some other queue (e.g., a cloud queue) might exist in the system beyond the claimed "*local playback queue*."

90. ***Second***, the '615 Patent contradicts Dr. Bhattacharjee's position and explains that multiple devices within the system can maintain a respective copy of a queue, where each respective queue copy contains all (or some portion) of the resource locators that correspond to what content is to be played back by at least one device within the system. In this way, multiple devices can share a playback queue. For example, the '615 Patent discloses that a playback device's "local playback queue" is kept "synchronized" with an "application-specific queue" maintained at a control device. '615 Patent, 16:20-31:

> [T]he third party application not only tells the local playback system what to play, but also maintains two-way communication with the local playback (e.g., Sonos™) system. Two-way communication helps enable features such as keeping a ***local playback queue*** synchronized with a ***queue*** that the user is editing/managing ***in the third party application***[.]

91. As another example, the '615 Patent describes how a playback device "periodically fetches a short list of tracks" from an "application-specific queue" that are then loaded into the playback device's "local playback queue." *See id.*, 16:63-17:1:

> Certain embodiments allow a third party application to ***override*** a ***local playback queue*** with its own ***application-specific queue***. The local playback system periodically fetches a short list of tracks to play next. The list of tracks to play is determined by the third-party application, for example.

92. As yet another example, the '615 Patent describes implementations in which a playback device's "local playback queue" and a control device's "application-specific queue" are synchronized to a "shared queue [ ] provided between" the two (e.g., a queue in the cloud). *See id.*, 16:63-17:4:

> Certain embodiments allow a third party application to override a ***local playback queue*** with its own ***application-specific queue***.... In certain embodiments, a ***shared queue*** is provided between the local playback system and the third party application to keep the local system and application synchronized.

93. ***Third***, as I explained before, Google's own documents describe multiple devices within Google's system sharing a queue much like the '615 Patent's teachings. *Supra* ¶¶54-57. In this way, Google's own words confirm that, although Casting involves cloud servers maintaining a queue (what Google is referring to as a "Cloud Queue"), the Sender and Receiver each maintains its own copy of the queue with at least some portion of the Cloud Queue's contents.

94. For at least these reasons, I disagree with Dr. Bhattacharjee's starting premise that a "Cloud Queue" and "*local playback queue*" are mutually exclusive and therefore, find Dr. Bhattacharjee's opinions to be fatally flawed.

b. Google's Receiver Does Not Retrieve Items "One-By-One"

95. For both YouTube and GPM, Dr. Bhattacharjee argues that a Receiver does not have a "*playback queue*" because it "retrieve[s] items from the Cloud Queue one-by-one." Bhatta. Decl., ¶¶83, 120. He is wrong.

96. For YouTube, Google's 30(b)(6) witness contradicted Dr. Bhattacharjee's argument and explained that there are circumstances where the Receiver's first received WNR provides the videoIds for the first two media items that the Receiver is to playback. Mo. Dep. Tr.,

158:3-7, 161:5-162:1, 163:4-164:1 ("[I]n the cast where there is no video ID, a watch next request is made with the playlist ID specifically so that the client can get a video ID. And then in response of watch next, there contains a variable or a field, something like a current watch endpoint" and "[i]t may contain some field that sounds like [a next endpoint]."). What's more, as I explained before, a WNR typically identifies multiple items from the Cloud Queue, such as an item before and after the current item. *Supra* ¶¶63-67; Mo. Dep. Tr., 166:19-22 (in response to "[c]an a watch next response contain two different video IDs, not a duplicate of the same video ID," answering "[y]es, it can."). For GPM, an IWR likewise contained multiple items from the Cloud Queue, including the current song, the song before, and the song after. *Supra* ¶¶84-85. Thus, the evidence contradicts the assertion that a Receiver retrieves items from the Cloud Queue "one-by-one."

c. A "Playback Queue" Need Not Include "All" Items

97. For both YouTube and GPM, Dr. Bhattacharjee argues that a Receiver must store the ***entirety*** of what the Receiver is scheduled to playback to have the claimed "*playback queue*." *See, e.g.*, Bhatta. Decl., ¶¶68, 72, 74, 115, 118. His position is another false premise contradicted by the '615 Patent itself and Google's own words.

98. As noted before, the '615 Patent teaches that, for example, a playback device "periodically fetches a short list of tracks" from a queue maintained at another device that are then loaded into the playback device's "local playback queue." *Supra* ¶¶90-92; '615 Patent, 16:63-17:1. In this way, a POSITA would readily appreciate that the '615 Patent expressly rejects Dr. Bhattacharjee's premise.

99. Moreover, Google's own admissions that I discussed before regarding its Receivers having a local queue despite the existence of a Cloud Queue in Google's system demonstrate that Dr. Bhattacharjee's position is incorrect. *Supra* ¶¶54-57.

100. Relatedly, Dr. Bhattacharjee makes an "offline" argument to support his assertion that the "YouTube system does not use a 'local playback queue'" (Bhatta. Decl., ¶¶74, 89), but this argument is premised again on the false assumption that the Receiver must store the entirety of what it is scheduled to be played back to have the claimed "*local playback queue*." Regardless, even in Dr. Bhattacharjee's hypothetical, the Receiver would know at least the respective videoIds

for the current and next media item to play (and perhaps also for the previous and/or "Autoplay" media items) and thus, would be able to continue playing those media items.

d. Media Items Need Not Be "Linked Together"

101. For both YouTube and GPM, Dr. Bhattacharjee argues that none of the accused data constructs amounts to a "*playback queue*" because a "*playback queue*" allegedly has to "link[] together different multimedia items in a particular order using linked lists, arrays, vectors, or other well-known data structures." Bhatta. Decl., ¶¶70, 82, 117.

102. But for the reasons I explained in my Claim Construction Report, I disagree that the '615 Patent limits a "*playback queue*" to any particular data construct, much less one that "link[s] … items in a particular order." *See* Dkt. 185-8, ¶¶74-90. In my opinion, such a requirement would be immaterial here because a Receiver stores in its memory multiple resource locators (e.g., videoIds for YouTube and itemID/trackUrl pairs for GPM) and updates its internal logic to know the order in which it is to use those resource locators to play corresponding content. In this regard, Google itself acknowledging that a Receiver maintains a local queue for playback in both the YouTube and GPM contexts (*supra* ¶¶54-57) – which implemented the Receiver's queue in different manners, as I explained above – confirms what I articulated in my Claim Construction Report:

> [A] POSITA would have appreciated that a "playback queue" is more about the general concept of defining what is to played back than a singularly defined type of data structure, and thus, a POSITA would have understood that a "playback queue" could be implemented in different ways and take different forms.

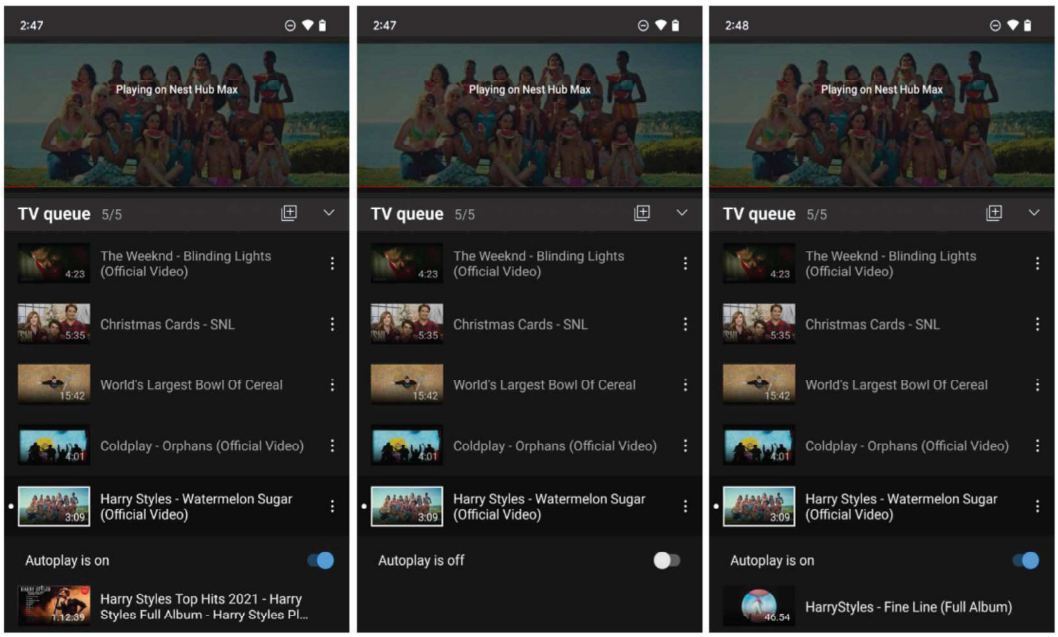
Dkt. 185-8, ¶87.

e. A User Can "Manage" a Receiver's "Playback Queue"

103. For both YouTube and GPM, Dr. Bhattacharjee argues that none of the accused data constructs amount to a "*playback queue*" because they are "fixed" and a "*playback queue*" allegedly must be able to be "managed" by a user. *See* Bhatta. Decl., ¶¶65, 70, 73, 84, 119. Dr. Bhattacharjee's argument is flawed for several reasons.

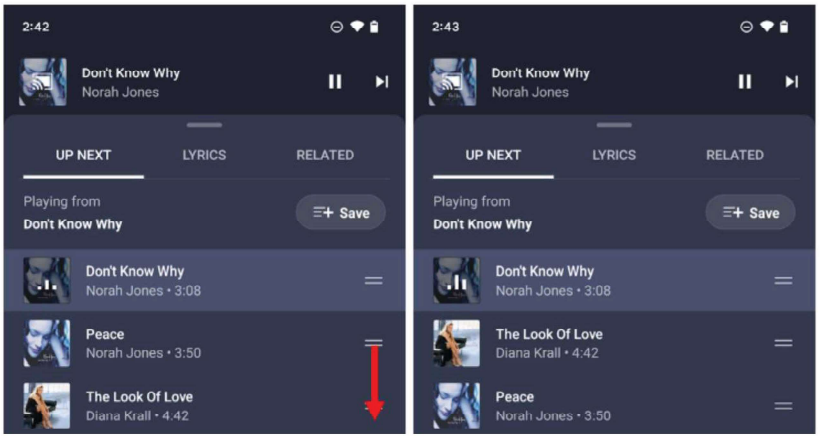
104. For starters, Dr. Bhattacharjee is reading into the claim term "*playback queue*" yet another limitation that is not required by '615 claim 13 or Google's narrow construction. Regardless, Dr. Bhattacharjee's argument is incorrect for the YouTube and GPM implementations.

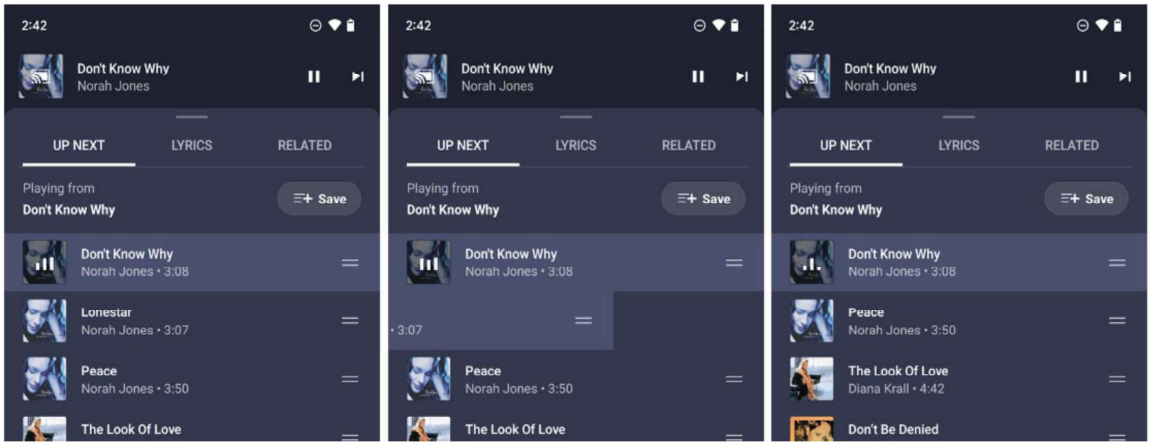
105. As to YouTube, the current and next videoId data variables, as well as the stored WNR, can indeed be updated to, for example, reflect what the Receiver is to play currently and play next. For instance, a user can manage the queue at the Sender, which in turn causes the videoIds of the Receiver's current and next data variables to be managed, as well. For example, as shown below, a user Casting YouTube to a Receiver can toggle the "Autoplay" setting at the Sender, which causes the Receiver's next videoId data variable to be updated by virtue of a new WNR. *See, e.g.*, Mo Dep. Tr., 178:17-179:10, 181:18-183:23; remote.ts[22] → updateAutoplayModeAndReloadWatchNext, lns. 2390-412 ("If the autoplay mode changed, notifies MDx and reloads watch next.").



106. As other examples, shown here, a user Casting YouTube Music can (i) modify the order of the songs such that the next song to play after the current song changes or (ii) delete the song that is scheduled to play next after the current song, which causes the Receiver's next videoId data variable to be updated. *See, e.g.*, remote.ts[23] → updatePlaylist, lns. 1522-29.

[22] /2021-02-01_YTReceivers09292020/google3/video/youtube/tv/bedrock/ts/mdx/
[23] /2021-02-01_YTReceivers09292020/google3/video/youtube/tv/bedrock/ts/mdx/





107. Similarly, GPM permitted the user to "manage" the Receiver's queue via the Sender, which in turn would have resulted in the stored itemID/trackUrl pair for one or more of the three multimedia items from the stored IWR to be updated accordingly. *See, e.g.*, Patil Dep. Tr., 88:3-9; playermanager.ts[24] → handleCloudQueueVersionResponse, lns. 2061-92.

f. Dr. Bhattacharjee's "Next" videoId Arguments Fail

108. Lastly, Dr. Bhattacharjee offers several specific arguments with respect to the "next" videoId data variable used in the YouTube context, but none of them are convincing.

109. For instance, while Dr. Bhattacharjee argues the data variable storing the next videoId is limited to "a recommended 'autoplay' video" that is only used once the "playlist is exhausted" (Bhatta. Decl., ¶¶71, 78-79), Google's 30(b)(6) witness explained that the videoId stored in the "next" data variable may or may not be a service-recommended (i.e., "Autoplay")

[24] /2021-01-27_GPMReceiver09292020/google3/java/com/google/wireless/android/skyjam/frontend/javascript/cast/

media item. *See* Mo Dep. Tr., 171:13-22 ("The watch next response could contain [fields] with video IDs that may differ. Again, the autoplay is one that could differ from the current. There could be – there is a notion of – there's autoplay set renderer where it will contain like next …."), 172:20-24, 174:10-175:23 (in response to "the next endpoint is not necessarily going to be an autoplay video; right," answering "[c]orrect, and in fact, the autoplay may not even be … it may not be labeled as the next video, they may just be also named like MDx autoplay."). In other words, the Receiver stores in its "next" data variable a videoId of the next media item that the Receiver is to play regardless of whether it is a service-recommended media item or not. Likewise, when playback of the current media item ends, the Receiver will look up that "data model" for the next media item and "extract the appropriate endpoint for the next [media item] to play." *Id.*, 175:1-23.

110. Moreover, there are numerous cases in which the second media item that the Receiver is to playback in a Cast session will be a service-recommended media item (e.g., whenever a user selects a single media item for playback using any of the YouTube apps before Casting (*supra* ¶59)). In those cases, the first WNR that is stored by the Receiver will contain a videoId for a service-recommended media item as the next videoId. *See* Mo Dep. Tr., 126:7-11 (in response to "[t]he autoplay recommended video would be the video that would be playing next after the user-selected video; correct," answering "[y]es, for the context where a single video is selected."), 127:22-128:1 ("For the specific case that I am thinking of, MDx autoplay comes from the watch next response …."), 178:17-179:10. Moreover, even outside of those use cases, the first WNR will often contain a videoId corresponding to a service-recommended media item that will be used once the "playlist is exhausted." *See id.*, 170:23-171:22 (explaining WNR "may contain the current watch video endpoint" and "could contain [fields] with video IDs that may differ. Again, the autoplay is one that could differ from the current. There could be – there is a notion of – there's autoplay set renderer where it will contain like next, previous, so forth, depending on the playback."), 174:10-16. Thus, Dr. Bhattacharjee's assertion that "the videoID for the recommended video is provided to the playback device only after the playlist is exhausted" (Bhatta. Decl., ¶78) is refuted by the evidence.

g. "Multimedia File" Need Not Be Added to "Local Playback Queue"

111. Claim 13 of the '615 Patent more fully recites:

> *causing one or more first cloud servers to add multimedia content to a local playback queue on the particular playback device,* **wherein adding** *the multimedia content to the local playback queue* **comprises** *the one or more first cloud servers adding, to the local playback queue, one or more resource locators corresponding to respective locations of the multimedia content at one or more second cloud servers of a streaming content service[.]*

112. Dr. Bhattacharjee argues that "*adding the multimedia content to the local playback queue*" requires adding "the actual multimedia file or information that is played back" to the "*local playback queue*." Bhatta. Decl., ¶¶90-94. I disagree that a POSITA would interpret claim 13 in this manner for several reasons.

113. ***First***, it is my opinion that a POSITA reading the plain language of claim 13 would understand that the "wherein" clause specifies ***how*** the "*one or more first cloud servers*" are to "*add multimedia content to a local playback queue*." In this respect, I have been informed that, in patent claims, (i) a "wherein" clause can provide meaning and purpose to a function that the clause modifies and (ii) the transitional phrase "comprising" or "comprises" is synonymous with "including," "containing," or "characterized by." This meaning is exactly what the "wherein" clause does here; it explains to a POSITA the meaning and characteristics of the function of the "*one or more first cloud servers [] add[ing] multimedia content to a local playback queue ….*" Specifically, the "wherein" clause explains that the claimed "*add[ing] multimedia content to a local playback queue*" means, and is characterized by, "*adding, to the local playback queue, one or more resource locators*" (as opposed to adding one or more multimedia files).

114. ***Second***, it is my opinion that the full context of claim 13 would also lead a POSITA to the inescapable conclusion that the "*one or more first cloud servers [] add multimedia content to a local playback queue*" **by** "*adding … one or more resource locators*" as opposed to adding one or more multimedia files. In particular, after "*causing one or more first cloud servers to add multimedia content to a local playback queue on the particular playback device*," claim 13 recites:

> *causing the particular playback device to play back the multimedia content, wherein the particular playback device playing back the multimedia content comprises the particular playback device* **retrieving the multimedia content from**

> **one or more second cloud servers** *of a streaming content service and playing back the retrieved multimedia content*.

A POSITA would readily appreciate that, if "*adding the multimedia content to the local playback queue*" was interpreted as Dr. Bhattacharjee views it to require adding one or more multimedia files to the "*local playback queue on the particular playback device*," there would be no need for the "*playback device*" to "*retriev[e] the multimedia content from one or more second cloud servers of a streaming content service*" because it would already have the multimedia content stored locally in its "*playback queue*." In my opinion, a POSITA would understand that such redundancy is not what the plain language of claim 13 requires.

115. ***Third***, a POSITA having read the '615 Specification would likewise understand that the claimed "*adding the multimedia content to the local playback queue*" refers to adding one or more "*resource locators*" (such as a URL or some other identification) to the "*local playback queue*," as opposed to adding one or more multimedia files. For instance, the '615 Patent states:

> [E]ach zone player 606, 604, 602 may access the Internet when retrieving media from the cloud (e.g., Internet) via the bridging device. For example, ***zone player 602 may contain a uniform resource locator (URL)*** that specifies an address to a particular audio track in the cloud. ***Using the URL, the zone player 602 may retrieve the audio track from the cloud***, and ultimately play the audio out of one or more zone players.

'615 Patent at 11:62-12:3. Likewise, the '615 Patent states:

> A uniform resource indicator (URI) (e.g., a uniform resource locator (URL)) can be passed to a playback device to fetch content from a cloud and/or other networked source, for example. A playback device, such as a zone player, can fetch content on its own without use of a controller, for example. Once the zone player has ***a URL (or some other identification or address) for a song*** and/or playlist, the zone player can run on its own to ***fetch the content***. Songs and/or other multimedia content can be retrieved from the Internet rather than a local device ….

Id. at 12:53-63. These exemplary passages demonstrate to a POSITA that a "*resource locator*" is first added to the playback device's "*playback queue*" before it retrieves the music corresponding to the "*resource locator*," which is what claim 13 recites.

116. ***Fourth***, the prosecution history confirms that Dr. Bhattacharjee's interpretation is inconsistent with a POSITA's understanding. In this regard, in its August 28, 2017 Office Action Response, Sonos distinguished the Togashi reference's disclosures by arguing that the "wherein"

clause modifying "*causing one or more first cloud servers to add multimedia content to a local playback queue on the particular playback device*" specifies ***how*** the "*one or more first cloud servers*" are to "*add multimedia content to a local playback queue*":

> Instead of requesting a content server to change the destination of the audio content [as in Togashi], Applicant's claims recite a different technique for transferring playback of multimedia content between devices. In particular, Applicant's claims recite "causing one or more first cloud servers to **add the multimedia content** to a local playback queue on the particular playback device" **by "adding,** to the local playback queue, **one or more resource locators** corresponding to respective locations of the multimedia content at the one or more second cloud servers of a streaming content service."

Dkt. 185-8, App'x B at 4 (original emphases omitted).

117. For at least these reasons, I disagree with Dr. Bhattacharjee's opinion that claim 13 requires adding a multimedia file to the "*local playback queue*." Thus, this cannot serve as a basis for non-infringement. Instead, the claims require "*one or more resource locators*" to be added to the "*local playback queue*," which is met by the accused instrumentalities, as explained below.[25]

h. Casting YouTube Involves Adding "Resource Locators"

118. As noted above, '615 claim 13 specifies that "*adding the multimedia content to the local playback queue*" is accomplished by adding "*one or more resource locators corresponding to respective locations of the multimedia content at one or more second cloud servers of a streaming content service*." In evaluating whether this element is satisfied, I applied the plain and ordinary meaning to "*resource locator*" in the context of the '615 Patent: information that enables a device to access a resource. *See* Dkt. 185-8, ¶101. I also applied the plain and ordinary meaning of "*corresponding to*": associated with or related to.

119. It is my opinion that the "*resource locator*" element is satisfied by the videoIds in (i) the current and next videoId data variables and (ii) WNR data structure that are stored by the Receiver because each videoId enables the Receiver to access (e.g., locate and retrieve) a particular media item from a Bandaid cloud server. *Supra* ¶68. Indeed, much like a PURL (which is a

[25] I note that Dr. Bhattacharjee did not make this "multimedia file" argument for GPM, yet I understand that Google did present this argument for GPM in Google's summary judgment motion. Regardless, for the same reasons as I explained in the YouTube context, this "multimedia file" argument cannot serve as a basis for non-infringement in the GPM context.

particular type of URL), the videoId in the YouTube system is provided to an intermediate service (the Player Service) that translates the videoId into at least one URL for the particular media item at a YouTube Bandaid server. *Id.* In fact, even one of Google's own engineers referred to the videoIds as "pointers" to content that the Receiver could play. *See* Nicholson Dep. Tr., 108:5-10 ("[A]n Autoplay renderer, which contains somewhat -- you might describe as pointers to other videos that the receiver could play next."), 114:19-115:1 ("For the next playing song, there's an Autoplay renderer…. [T]hat Autoplay renderer … will contain information about what video ID should the device play next if the user moves into the next song."). Thus, Dr. Bhattacharjee's opinion that a videoId "does not provide any information about the location where the video is stored" is flawed. Bhatta. Decl., ¶¶96-97.

120. Moreover, it is my opinion that the "*resource locator*" element is satisfied even under Google's narrow construction: an address of a resource on the Internet. For instance, as noted before, by virtue of causing the MDx and/or WatchNext servers to add one or more videoIds to the Receiver's "*local playback queue*," the Sender also causes the Player Service server to add one or more URLs for each videoId to the Receiver's memory. *Supra* ¶68. In particular, the URLs are saved in a data variable called dashManifest. *See, e.g.*, video_data.ts[26]→ dashManifest, lns. 452-53; loadDashMpd, lns. 3289-344; onDashMpdLoadSuccess, lns. 3485-93.

121. It is my opinion that this functionality would literally satisfy the "*resource locator*" element under Google's construction because the Sender causes the Player Service to add one or more URLs (which Google admits are "*resource locators*") for each videoId to the Receiver's memory (i.e., in the dashManifest data variable), which would constitute the "*local playback queue*" in this alternative infringement theory because it is a data construct on the Receiver that contains a resource locator (here, a URL) corresponding to multimedia content the Receiver is to playback.

122. It is also my opinion that this functionality would satisfy the "*resource locator*" element under Google's construction under DoE because there is merely an insubstantial

[26] /2021-02-01_YTReceivers09292020/google3/video/youtube/web/player/

difference between a Sender (i) causing one or more cloud servers to add at least one "address of a resource on the Internet" to a "*local playback queue*" and (ii) causing one or more cloud servers to add at least one videoId to a "*local playback queue*" and another cloud server to subsequently translate the videoId into an "address of a resource on the Internet" that the Receiver uses for playback. In fact, whether a Sender directly or indirectly causes one or more cloud servers to add at least one "address of a resource on the Internet," the Sender is performing substantially the same function (e.g., causing one or more cloud servers to add multimedia content to the Receiver's "*local playback queue*"), in substantially the same way (e.g., by sending a message to the one or more cloud servers that causes this function to occur without any intervening user action), to achieve substantially the same result (e.g., one or more cloud servers adding at least one "address of a resource on the Internet" to the Receiver's memory to facilitate playback). In other words, it is my opinion that there is merely an insubstantial difference between a cloud server adding an "address of a resource on the Internet" in one step and a cloud server adding information representing an "address of a resource on the Internet" in a first step that is resolved into that address in a subsequent step.

VIII. VALIDITY ANALYSIS

A. Google's Invalidity Theories

123. As set forth in Dr. Bhattacharjee's declaration, Google asserts that '615 claim 13 is anticipated by the YouTube Remote ("YTR") software application installed on a mobile phone, which Google asserts as "system" prior art. *See* Bhatta. Decl., ¶¶121-23. Alternatively, Google asserts that '615 claim 13 is obvious based on the YTR application in view of (i) the general knowledge of a POSITA and/or (ii) U.S. Patent No. 9,490,998 (the "'998 Patent"). *Id.*

B. Invention Date

124. I have been informed that Sonos has asserted an invention date for the '615 Patent of July 15, 2011. I note that, for purposes of summary judgment, Google is not challenging the July 15, 2011 invention date. *See id.*, ¶124.

C. Alleged Prior Art References and State of the Art

1. Overview of the YouTube Remote Application

125. According to a YouTube press release, a "beta" version of the YTR application for mobile phones running the Android operating system appears to have been released on November 9, 2010. GOOG-SONOS-WDTX-INV-00015413, 13. As explained in the title of the press release, the YTR application could be used to "[c]ontrol YouTube on the desktop, or the TV." *Id.* To enable this functionality, the YTR application "create[d] a virtual connection between your phone and YouTube Leanback" on a "Google TV or computer." *Id.* The connected TV or desktop device was referred to as a "Leanback screen" or just "screen." *Id.* Herein I refer to such as device as a "Leanback Screen."

126. As explained in the press release: "To 'pair' your phone with your Leanback screen, simply sign into YouTube Remote on your Android phone, and to YouTube Leanback on your Google TV or computer with the same YouTube account. Just like that, you've connected your powerful multi-touch Android screen with the biggest screen in your home." *Id.* I understand that the pairing of the YTR application with a Leanback Screen was facilitated by an intermediary YouTube cloud server referred to by Google as a "Lounge Server" or "MDx server." *See* Bhatta. Decl., ¶¶128, 132.

127. After a "virtual connection" (or "pairing") was established, I understand that a user could then use the YTR application to control playback of videos on the Leanback Screen. GOOG-SONOS-WDTX-INV-00015413, 13-14. To the extent the YTR application could be paired with multiple Leanback Screens in a session, it is my understanding that the same media would be played on ***all*** the Leanback Screens and that the YTR application and/or cloud-based Lounge Server would be configured to control the playback on all paired Leanback Screens ***collectively*** (as opposed to controlling any one of the multiple Leanback Screens ***individually*** or controlling a subset of the multiple Leanback Screens). *See* Bhatta. Decl., ¶¶140, 144, 158, 160-62; GOOG-SONOS-WDTX-INV-00015423, 24.

128. As discussed in Video #2 cited by Dr. Bhattacharjee, the connection between a YTR application and a Leanback Screen could be established when a mobile phone installed with the YTR application was connected to a local area network (LAN), such as a home Wi-Fi network, or when it was only connected to a wide area network (WAN), such as a 3G cellular network. *See*

Levai Dep. Tr. at 94:2-21; https://www.youtube.com/watch?v=5VFIuR9pJdo (GOOG-SONOSNDCA-00071320), 2:51-3:20 (explaining that a mobile phone running YTR application can connect to and control a Leanback Screen "***over 3G and Wi-Fi***"); *see also* Bhatta. Decl., ¶157 ("In order to pair with one another a mobile device running the YTR application and a Screen both had to be connected to the ***Internet***—which ***could be*** done by connecting to a user's home network [a 'local area network'] through Wi-Fi."). I understand that the ability to establish the connection regardless of what type of network the mobile phone was connected to was facilitated by the cloud- or WAN-based Lounge Server, which acted as an intermediary between the mobile phone running the YTR application and the Leanback Screen. *Id.*; GOOG-SONOSWDTX-00041837; *see also* '998 Patent, 4:51-55 ("[B]y using the network service as an intermediary, the remote control and the controlled device … may ***not need to be connected to the same local area network***, nor in physical proximity to each other.").

129. In his declaration, Dr. Bhattacharjee attempts to explain various aspects of the YTR application. However, as explained below in connection with my anticipation and/or obviousness analysis, I disagree with many of Dr. Bhattacharjee's characterizations.

130. I also disagree with many of Dr. Bhattacharjee's assertions that various materials/references he relies on evidence the ***same*** November 9, 2010 version of the YTR application and/or are actually prior art to the '615 Patent. For example, Dr. Bhattacharjee initially points to what he calls "Version 1.0" of the YTR application that was allegedly released on November 9, 2010 and allegedly shown in various videos that were allegedly published between November 9, 2010 and February 14, 2011. *See* Bhatta. Decl., ¶¶124-25. However, Dr. Bhattacharjee then goes on to rely on various source code for the YTR application, including source code that allegedly "reflects the operation of the YTR system that was released on November 9, 2010," as well as "YTR system source code as it existed on July 12, 2011" and "YTR system source code as it existed on December 1, 2011." *Id.*, ¶126. By Dr. Bhattacharjee's own admission, the 2011 source code "is for subsequent releases of the YTR application." *Id.* Thus, the 2011 source code is not evidence of the operation of Version 1.0 of the YTR application that was allegedly released on November 9, 2010, which is the version that Google is relying on for

invalidity in its summary judgment motion. *Id.*

131. Moreover, contrary to Dr. Bhattacharjee's suggestion, the December 1, 2011 source code is not even prior art to the '615 Patent because it is dated ***after*** the July 15, 2011 invention date asserted by Sonos and unchallenged by Google in its summary judgment motion. And while Dr. Bhattacharjee asserted that some of the source code he reviewed "reflects the operation of the YTR system that was released on November 9, 2010," I have seen no evidence substantiating that date. Instead, Dr. Bhattacharjee appears to rely solely on the uncorroborated Declaration of Janos Levai. *See* Bhatta. Decl., ¶126 (citing Levai Decl., ¶6). However, Mr. Levai did not start working at Google until he joined as an intern in July 2011. Levai Decl., ¶2. Thus, it is unclear how Mr. Levai can definitively establish that certain source code reviewed by Dr. Bhattacharjee corresponds to Version 1.0 of the YTR application that was allegedly released on November 9, 2010. Notably, another one of Google's declarants, Ramona Bobohalma, did ***not*** verify that any of the source code reviewed by Dr. Bhattacharjee corresponds to Version 1.0 of the YTR application that was allegedly released on November 9, 2010. This lack of verification is despite the fact that Ms. Bobohalma actually worked at Google in 2010 at the time Version 1.0 of the YTR application was allegedly released and "was responsible for the development of Versions 1 through 3 of Google's Multi-Device Experience ('MDx') protocol, including writing source code for MDx." Bobohalma Decl., ¶¶2-3. I also understand that the source code Dr. Bhattacharjee relies on for the operation of the YTR system on November 9, 2010 is actually an alleged "capture of the code that existed on November 11, 2010" despite the fact that Google apparently could have obtained a November 9, 2010 capture of the code. *See* Levai Decl., ¶6; Levai Dep. Tr., 64:21-65:2. Mr. Levai also stated that he didn't know whether any changes were made to the code between November 9, 2010 and November 11, 2010. *Id.* at 66:4-13.

132. Moreover, I have not seen any evidence definitively establishing that each of the six different videos Dr. Bhattacharjee cites to shows the operation of Version 1.0 of the YTR application. *See* GOOG-SONOS-WDTX-INV-00015101 ("Video #1" uploaded November 14, 2010); GOOG-SONOSNDCA-00071320 ("Video #2" uploaded November 11, 2010); GOOG-SONOSNDCA-00071319 ("Video #3" uploaded November 9, 2010); GOOG-SONOSNDCA-

00015102 ("Video #4" uploaded November 9, 2010); GOOG-SONOSNDCA-00071317 ("Video #5" uploaded November 15, 2010); GOOG-SONOSNDCA-00071318 ("Video #6" uploaded February 14, 2011). Likewise, I have also not seen any evidence definitively establishing that the YTR applications allegedly shown in these six different videos are all the same version of the YTR application. In that regard, both Mr. Levai and Ms. Bobohalma appear to implicitly recognize that Video #6 does not accurately show the capabilities of Version 1.0 of the YTR application that was allegedly released on November 9, 2010, as Video #6 was not addressed by either declarant. *See* Levai Decl., ¶5; Bobohalma Decl., ¶5.

133. As another example, Dr. Bhattacharjee relies on screenshots in a February 26, 2012 Wayback Machine capture of a version of the YTR application that appears to have been released on January 25, 2012. *See* Bhatta. Decl., ¶170 (citing Google MSJ Ex. 14). This version of the YTR application is not prior art to the '615 Patent because it is dated ***after*** the July 15, 2011 invention date asserted by Sonos and unchallenged by Google in its summary judgment motion. Additionally, it is unclear if the version of the YTR application that Google relies on was actually available in the United States because the webpage captured by the Wayback Machine includes non-English language. *Id.*

134. As yet another example, Dr. Bhattacharjee relies on an "API" document that is dated July 12, 2010. *See* Bhatta. Decl., ¶137 (citing GOOG-SONOSNDCA-00056724). However, Dr. Bhattacharjee has not established that this API document (or the specific disclosure that he relies on) reflects the operation of Version 1.0 of the YTR application that was released some four months later on November 9, 2010. Instead, the only evidence relied on by Dr. Bhattacharjee appears to be the uncorroborated declaration of Mr. Levai stating that "[t]he document generally describes the API for the first release of the YouTube Remote application." Levai Decl., ¶9. Again, Mr. Levai did not even work at Google in 2010.

2. Overview of the '998 Patent

135. Google relies on the '998 Patent to support both of its obviousness theories, namely, (i) to establish the state of the art and the "general knowledge" that a POSITA would have used to modify the YTR application and (ii) as a secondary reference to combine with the YTR application.

136. The '998 Patent is entitled "Network-Based Remote Control" and issued from an application filed on March 7, 2011, which claims priority to November 8, 2010. The '998 Patent lists Google as the assignee. One of the named inventors of the '998 Patent is Ms. Bobohalma, who stated that "[t]his patent discloses ***some*** of the work that I did on the YouTube Remote application." Bobohalma Decl., ¶4.

137. Consistent with the above-described November 2010 YouTube press release for the YTR application, the '998 Patent discloses a "remote control" that can be "paired" with a "controlled device" having a screen and thereafter used to control playback of media content on the "controlled device," "such as stopping playback of media content playing on the controlled devices or changing the media content playing on the controlled devices." *See, e.g.*, '998 Patent, 1:39-46, 3:34-55, Fig. 1. The '998 Patent does ***not*** disclose initiating playback of media on the "remote control" and then transferring playback from the "remote control" to a "controlled device." The general architecture of the '998 Patent's system is shown in Fig. 1 here.

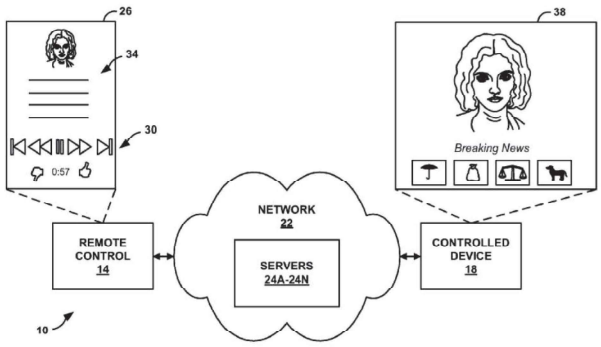


FIG. 1

138. Like the November 2010 YouTube press release for the YTR application, the '998 Patent discloses a "pairing" technique that utilizes a "network service" (also referred to as a "cloud service") provided by a server(s) that acts as an "intermediary" between the "remote control" and "controlled device." *See, e.g.*, *id.*, 3:13-45, Fig. 1. For example, the '998 Patent discloses that "a user may log in to a user account maintained by the servers 24 using remote control 14" and that the "[r]emote control 14 may … transmit a message to servers 24 of network 22 that identifies remote control 14, which can be used by servers 24 to pair remote control 14 with controlled device 18." *Id.*, 5:19-26. Similarly, the '998 Patent discloses that "a user may log in to a user account maintained by the servers 24 using controlled device 18" and that "[c]ontrolled device 18 can … transmit a message to servers 24 of network 22 that identifies controlled device 18, which can be used by servers 24 to pair controlled device 18 with remote control 14." *Id.*, 5:53-60.

139. This intermediary "cloud service" architecture allows the "remote control" and "controlled device" to be paired together and communicate with each other when, for example, the two devices are not on the same local area network. *Id.*, 4:51-55 ("[B]y using the network service as an intermediary, the remote control and the controlled device … may ***not need to be connected to the same local area network***, nor in physical proximity to each other.").

140. Notably, the '998 Patent appears to distinguish its system architecture where the "cloud service" serves as an intermediary for both pairing and subsequent communication between a "remote control" and "controlled device" from prior art systems where remote controls communicate directly with the devices being controlled. *Id.*, 1:14-35.

141. In his declaration, Dr. Bhattacharjee attempts to explain various aspects of the system disclosed in the '998 Patent. However, as explained below in connection with my anticipation and/or obviousness analysis, I disagree with many of Dr. Bhattacharjee's characterizations of the '998 Patent.

3. Overview of the Tungsten System, the Apple Airplay System, U.S. Publication No. 2011/0131520 ("Al-Shaykh Publication"), & Sonos's Products

142. Dr. Bhattacharjee relies on the Tungsten system, the Apple Airplay system, the Al-Shaykh Publication, and Sonos's products to support his obviousness assertions regarding the state of the art and the "general knowledge" that a POSITA would have used to modify the YTR application. In sum, Dr. Bhattacharjee relies on these systems/references for alleged teachings of "the ability to select individual playback devices for transfer from the user interface of a mobile phone or tablet." Bhatta. Decl., ¶¶25-28, 30-34, 166-167, 173. However, Dr. Bhattacharjee cites very little evidence regarding these systems/references and does not describe the specific system architecture used or disclosed by these systems/references or explain if or how the control devices of such systems identify playback devices connected to the same local area network as the control devices. *Id.*

143. As explained below in connection with my anticipation and/or obviousness analysis, I disagree with many of Dr. Bhattacharjee's characterizations of these systems/references.

144. I also understand that Apple TV manuals that describe Apple's Airplay functionality and Sonos User Guides were thoroughly reviewed and considered by the USPTO during prosecution of the '615 Patent, and the UPSTO allowed the '615 Patent (including the asserted claims) to issue over those Apple TV manuals. As a result, I understand that with respect to Dr. Bhattacharjee's invalidity arguments based on the functionality of Apple's Airplay and/or Sonos's products, Dr. Bhattacharjee has the added burden of overcoming the deference that is due to a qualified government agency, such as the USPTO, that is presumed to have properly done its job based on its expertise in interpreting references, its understanding of the level of ordinary skill in the art, and its duty to issue only valid patents.

D. Anticipation Analysis

145. In my opinion, the YTR application does not anticipate claim 13 of the '615 Patent because it fails to disclose at least claim limitations 13.2 and 13.4.

1. The YTR Application Does Not Disclose Limitation 13.2

146. Limitation 13.2 requires "*after connecting to a local area network via a network interface, identifying playback devices connected to the local area network.*" The plain language of this limitation requires the claimed "*control device*" to not only ***identify*** playback devices, but also to identify playback devices that are connected to the ***same local area network*** as the control device. Moreover, when properly interpreted in view of the surrounding claim language, especially limitation 13.4, it is clear that the "*identifying*" required by limitation 13.2 must allow a particular playback device from the identified playback devices to be subsequently selected via the control device to transfer playback of multimedia from the control device to the particular playback device. *See* limitation 13.4 ("*detecting a set of inputs to transfer playback from the control device to a particular playback device, wherein the set of inputs comprises: ... (ii) a selection of the particular playback device from the identified playback devices connected to the local area network*"). The YTR application does not teach these requirements.

147. According to Dr. Bhattacharjee, the YTR application performs the required "*identifying*" of limitation 13.2 because "each time a Screen on a local area network registers with the YT Lounge Server, the YT Lounge Server sends the YTR application a

'loungeScreenConnected' message." Bhatta.Decl., ¶158; GOOG-SONOSWDTX-00041837, 37. I disagree. As explained in the primary document relied upon by Dr. Bhattacharjee, the "loungeScreenConnected" message merely "informs the [YTR application] that there is at least one screen connected in the session."[27] *Id.* Merely notifying the YTR application that there is "***at least one screen*** … in the session" (and doing so via a message sent from the ***cloud-*** or ***WAN-***based Lounge Server) does not enable the YTR application to ***identify*** the Leanback Screen in the session, let alone identify the Leanback Screen as being connected to the ***same LAN*** that the mobile phone running the YTR application is connected to, ass required by limitation 13.2. *See id.* Likewise, as explained more below with respect to limitation 13.4, this inability of the YTR application to identify the Leanback Screens that are in the session prevents a particular Leanback Screen from subsequently being presented by the YTR application for selection in order to transfer playback from the YTR application to the particular Leanback Screen.

148. Moreover, even if the "loungeScreenConnected" message did include information from which the YTR application could identify the specific Leanback Screen that was registered with the Lounge Server and thus in a "session" (I have seen no such evidence), this would still not satisfy limitation 13.2 because the YTR application could still not identify the Leanback Screen as being connected to the ***same local area network*** that the mobile phone running the YTR application is connected to. This failure to satisfy limitation 13.2 makes sense because the YTR application receives the "loungeScreenConnected" message from the ***cloud***-based Lounge Server regardless of whether the mobile phone running the YTR application and the Leanback Screen are both operating on the same local area network or different local area networks, or whether the mobile phone running the YTR application is only operating on a wide area network (e.g., a 3G cellular network), and the "loungeScreenConnected" message provides no indication of the

[27] The use of the term "Connected" in the name of this message refers to the Leanback Screen being registered with and connected to the cloud-based Lounge Server and thus, in a "session" where the Leanback Screen can be controlled by a YTR application – it does ***not*** refer to a connection between the Leanback Screen and whatever network (e.g., local area network or wide area network) the Leanback Screen and/or mobile phone running the YTR application is operating on. *See* GOOG-SONOSWDTX-00041837, 37.

network that the Leanback Screen is connected to. *See* GOOG-SONOSWDTX-00041837, 37; *see also* Levai Dep. Tr. at 94:2-21; GOOG-SONOSNDCA-00071320, 2:51-3:20 (explaining that a mobile phone running YTR application can connect to and control a Leanback Screen "***over 3G and Wi-Fi***"); Bhatta. Decl., ¶157 ("In order to pair with one another a mobile device running the YTR application and a Screen both had to be connected to the ***Internet***—which ***could be*** done by connecting to a user's home network … through Wi-Fi."); '998 Patent, 4:51-55 ("[B]y using the network service as an intermediary, the remote control and the controlled device, in various instances, may not ***need to be connected to the same local area network***, nor in physical proximity to each other.").

149. Thus, because the YTR application fails to teach limitation 13.2, it does not anticipate claim 13 of the '615 Patent.

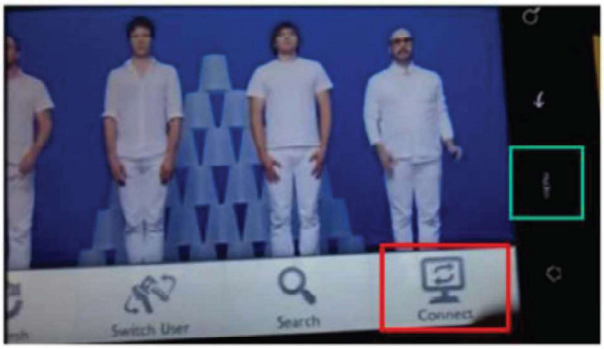
2. The YTR Application Does Not Disclose Limitation 13.4

150. Limitation 13.4 requires:

> *detecting a set of inputs to transfer playback from the control device to a particular playback device, wherein the set of inputs comprises: (i) a selection of the selectable option for transferring playback from the control device and (ii) a selection of the particular playback device from the identified playback devices connected to the local area network.*

The plain language of this limitation requires at least ***two separate and distinct inputs*** to transfer playback from the control device to a particular playback device, where each input must meet the specific requirements set forth in limitation 13.4, subparts (i) and (ii). The YTR application does not teach the two required inputs of this limitation.

151. According to Dr. Bhattacharjee, the YTR application detects the required inputs of limitation 13.4 via selection of the "menu" and/or "Connect" buttons shown in the image here. *See* Bhatta. Decl., ¶¶159-63 (citing GOOG-SONOS-WDTX-INV-00015101 ("Video #1")) (annotations in original). I disagree.



152. Selection of the "menu" button does not meet either of the required inputs because,

for example, it is not a button for transferring playback from the YTR application or a button for selecting a particular playback device to transfer playback to. Instead, the "menu" button does exactly what its name describes, it simply activates a "menu" on the YTR application, where the "menu" includes four different selectable buttons: a "Refresh" button, a "Switch User" button, a "Search" button, and a "Connect" button.

153. The only other input identified by Dr. Bhattacharjee is the input received via selection of the "Connect" button. *See* Bhatta. Decl., ¶¶159-63.[28] However, contrary to Dr. Bhattacharjee's assertion, a ***single*** input via the "Connect" button cannot satisfy the ***two*** separate and distinct inputs required by limitation 13.4. *Id.*, ¶¶161-62. Thus, the YTR application fails to teach limitation 13.4.

154. Moreover, even if a single input could somehow satisfy the two separate and distinct inputs of claim 13 (it cannot), I disagree with Dr. Bhattacharjee that a selection of the "Connect" button is "*a selection of* ***the particular playback device*** *from the identified playback devices connected to the local area network*," as required by limitation 13.4. *See* Bhatta. Decl., ¶162. For instance, as shown in the image above, the "Connect" button presented by the YTR application does ***not*** provide any indication of ***the particular Leanback Screen*** to transfer multimedia to. This makes sense because, as discussed for limitation 13.2, the YTR application does not even identify the specific Leanback Screens that are paired in a session with the YTR application. Without such an identification, the YTR application cannot present a particular Leanback Screen for selection or detect its selection.

155. Further, Dr. Bhattacharjee's own argument confirms that a selection of the "Connect" button is ***not*** "*a selection of* ***the particular playback device*** *from the identified playback devices connected to the local area network*," as required by limitation 13.4. According to Dr.

[28] Notably, limitation 13.5 requires the "*transferring playback*" to include "*causing the playback at the control device to be* ***stopped***." However, the November 14, 2010 video relied on by Dr. Bhattacharjee does not establish that the media on the YTR device was "*stopped*" after the "Connect" button was selected. *See* GOOG-SONOS-WDTX-INV-00015101 ("Video #1"). Although unclear in the video, the media on the phone's screen appears to still be in a playback state (albiet paused) after the selection of the "Connect" button and the alleged "*transfer*" of playback.

Bhattacharjee, in a scenario where multiple Leanback Screens are paired in a session with a YTR application, "[p]ressing the Connect button transfers playback to ***all the Screens*** that have been paired with the YTR application in a session …." *Id.* In other words, from the perspective of the YTR application, a selection of the "Connect" button merely indicates a selection of ***all*** Leanback Screens in a session. A selection of a button divorced from any particular Leanback Screen is ***not*** a selection of "***the particular***" Leanback Screen to transfer playback to.[29]

156. Further still, Dr. Bhattacharjee's own allegations regarding the development of the YTR application confirm that the alleged November 9, 2010 prior art version of the YTR application is missing features of limitation 13.4. Specifically, Dr. Bhattacharjee asserts that Google "released" a later ***non-prior art*** version of the YTR application on January 25, 2012 that, unlike the alleged prior art version, included the ability to "select and control" "individual" Leanback Screens for playback/control. *See* Bhatta. Decl., ¶¶169-70.[30]

157. Thus, because the YTR application fails to teach limitation 13.4, it does not anticipate claim 13 of the of the '615 Patent.

E. Obviousness Analysis

158. As noted above, Google is also asserting that '615 claim 13 is obvious based on the YTR application in view of (i) the "general knowledge" of a POSITA and/or (ii) the '998 Patent. *See* Bhatta. Decl., ¶¶121-23. While Dr. Bhattacharjee provides an analysis of obviousness in the section of his declaration for limitation 13.4, he appears to also be asserting that limitation 13.2 is obvious. Regardless, below I address Dr. Bhattacharjee's obviousness assertions and explain why neither limitation 13.2 nor limitation 13.4 would have been obvious.

1. The YTR Application + General Knowledge of a POSITA

---

[29] Consistent with Dr. Bhattacharjee's assertion, my understanding is that, to the extent multiple Leanback Screens can be paired with a YTR application in a single session, the YTR application can only be used to control playback on ***all*** of the Leanback Screens in the session ***collectively*** – it cannot be used to control playback on one of multiple Leanback Screens ***individually*** or control a subset of the multiple Leanback Screens. *Supra* ¶127.

[30] Dr. Bhattacharjee also asserts that this added functionality was included in a December 1, 2011 capture of source code (*see* Bhatta. Decl., ¶170) but that is irrelevant because it is also after Sonos's uncontested July 15, 2011 invention date.

159. Dr. Bhattacharjee appears to be arguing that it would have been obvious to modify the YTR application to include the requirements of limitations 13.2 and 13.4 because "[a] POSITA would have understood that … the YTR application[] would benefit from being able to (1) identify playback devices available for transfer on the local area network and (2) display them on the user interface of the mobile device for selection by the user." Bhatta. Decl., ¶165. Dr. Bhattacharjee further asserts that such identification and display were allegedly "well-known" to a POSITA prior to the July 15, 2011 invention date (e.g., by virtue of the Tungsten system, the Apple AirPlay system, the Al-Shaykh Publication, the '998 Patent, and Sonos's products) and that these features would have been "straightforward" to implement in the YTR application that allegedly existed prior to the July 15, 2011 invention date. *Id.*, ¶¶166-71, 173-74. I disagree.[31]

160. For instance, as an initial matter, Dr. Bhattacharjee has not cited any evidence showing that the Tungsten system, the Al-Shaykh Publication, the '998 Patent, or Sonos's alleged prior art products (all of which allegedly represent the "general knowledge" of a POSITA) enabled or disclosed "***transferring playback***" of media from a control device to a playback device as that term is used in claim 13 of the '615 Patent, let alone any "*inputs to* ***transfer playback*** *from the control device to a particular playback device*," including "*a selection of the particular playback device from the identified playback devices connected to the local area network*," as required by limitation 13.4. *Id.*, ¶¶26-34. Instead, the cited evidence merely shows that a control device could be used for ***controlling playback*** on one or more playback devices (e.g., starting or stopping playback) or for ***streaming media content*** from the control device to a playback device. *Id.* In the context of claim 13, "*transferring playback*" from a "*control device*" to a "*particular playback device*" is not met by merely streaming media content from a control device to a playback device and/or initiating playback on one or both devices. *See* limitations 13.5-13.6. The control device has to be capable of being in a playback state when it "*detect[s] a set of inputs to transfer playback*" and after such detecting, "*caus[e] playback to be transferred from the control device to the*

[31] I address specific failures with respect to the '998 Patent below in connection with Dr. Bhattacharjee's obviousness theory based on the combination of the YTR application and the '998 Patent. However, those failures also apply here to Dr. Bhattacharjee's theory based on the "general knowledge" of a POSITA.

particular playback device," which includes "*causing playback at the control device to be stopped*" and "*causing the particular playback device to play back the multimedia content*." *Id.* Thus, contrary to Dr. Bhattacharjee's assertion, a POSITA would not have been motivated to seek out or draw upon their "general knowledge" about these alleged prior art references to modify the YTR application to include the features of limitation 13.4, which are specifically directed to "*inputs to **transfer playback** from the control device to a particular playback device*."

161. Likewise, I disagree with Dr. Bhattacharjee's assertion that a POSITA would have been motivated to seek out or draw upon their "general knowledge" about the alleged Apple AirPlay prior art system to modify the YTR application to include the features of limitations 13.2 and/or 13.4. Based on the limited Apple AirPlay evidence cited by Dr. Bhattacharjee, it appears that Apple AirPlay utilized a fundamentally different system architecture than the YTR application that did not rely on an intermediary cloud server to, for example, "pair" control devices with playback devices or facilitate communications between the devices even when the devices were not on the same local area network. Instead, as explained in the "Airplay Video #1" cited by Dr. Bhattacharjee, Apple AirPlay "works automatically, through your Wi-Fi network, with no setup needed." *See* https://www.youtube.com/watch?v=fGMdg13YWB8, 0:25 - 0:29. In this regard, I understand that an AirPlay control device was only able to control an AirPlay playback device if both devices were on the ***same*** Wi-Fi network. *Id.* However, as explained above, the YTR application was specifically designed with only a ***cloud***-based communication channel to allow the YTR application to be paired with and control a Leanback Screen even if the mobile phone running the YTR application was ***not on the same*** Wi-Fi network as the Leanback Screen. *Supra* ¶128. Thus, contrary to Dr. Bhattacharjee's assertion, a POSITA would not have been motivated to modify the YTR application, which allowed the YTR application to be paired with and control Leanback Screens on different networks, with a system like AirPlay, which required the control device and players to be on the same network. Likewise, Dr. Bhattacharjee has not explained how a POSITA would have modified the YTR application in a manner that would have allowed for the YTR application to identify and present for selection Leanback Screens on the same local area network as the mobile phone while maintaining the touted ability for the YTR application to be

paired with and control a Leanback Screen via the mobile phone's 3G cellular network connection.

162. Moreover, in the '998 Patent, which I understand to disclose "some" features of the YTR application (*see* Bobohalma Decl., ¶4; Bhatta. Decl., ¶29), Google teaches away from modifying the YTR application to incorporate features of a system like Apple AirPlay when it distinguished the system architecture of the YTR application that relies on the intermediary cloud-based Lounge Server from system architectures of other prior art systems that enabled remote controls to communicate directly with playback devices via, for example, a home Wi-Fi network. *See* '998 Patent, 1:14-50; GOOG-SONOSNDCA-00075593, 96 (Google characterizing YTR as "fundamentally different" than Apple AirPlay).

163. Due to these fundamental differences between the system architecture of the YTR application and the architecture of a system like Apple AirPlay, which Google itself has acknowledged, it is my opinion that a POSITA would not have been motivated to modify the YTR application in view of Apple AirPlay.

164. As another example, the system architecture supporting the functionality of the alleged prior art version of the YTR application teaches away from modifying the YTR application in the manner proposed by Dr. Bhattacharjee. Moreover, at the very least, the architecture makes such modification harder than Dr. Bhattacharjee asserts and would require changes that Dr. Bhattacharjee fails to address in his conclusory analysis. As explained by Dr. Bhattacharjee, "[a] YTR application and one or more Screens are paired together by [a user] logging into the same YouTube account [on both devices] which registers them with the Lounge Server and adds them to a 'session.'" Bhatta. Decl., ¶132; *see also* GOOG-SONOS-WDTX-INV-00015413, 13 ("To 'pair' your phone with your Leanback screen, simply sign into YouTube Remote on your Android phone, and to YouTube Leanback on your Google TV or computer with the same YouTube account."). In other words, when the user wished to control media playback on a particular Leanback Screen using the YTR application, the user identified the particular Leanback Screen they desired to control (without caring whether the Leanback Screen and YTR application were connected to the same local area network) and used the ***user interface of that particular Leanback Screen*** to log the Leanback Screen into the same YouTube account that the YTR application was

logged into (or subsequently would be logged into). This capability is what caused the cloud-based Lounge Server to pair the devices in a session. *See* Bhatta. Decl., ¶¶128, 132. Given this architecture, there was no need for the YTR application to identify the Leanback Screen, let alone identify it as a Leanback Screen on the same local area network as the YTR application, and then present the Leanback Screen for selection via ***the user interface of the YTR application*** because the user had already identified and selected the particular Leanback Screen that the user wished to control. In this regard, the YTR application taught against detecting the claimed "selection" required by limitation 13.4, subpart (ii). Thus, a POSITA would not have been motivated to modify the YTR application in this manner, as proposed by Dr. Bhattacharjee.

165. Moreover, my understanding is that, in the event a YTR application and multiple Leanback Screens were paired together in a session by the Lounge Server (which is the scenario Dr. Bhattacharjee primarily relies on for invalidity), the YTR application and/or Lounge Server were configured to control the playback of the same media on ***all*** paired Leanback Screens ***collectively***. *See* Bhatta. Decl., ¶¶140, 144, 158, 160-162; GOOG-SONOS-WDTX-INV-00015423, 24. When multiple Leanback Screens are in a session, I am not aware of any disclosure in the materials cited by Dr. Bhattacharjee of the YTR application and/or Lounge Server having the capability to control ***a particular*** paired Leanback Screen. While Dr. Bhattacharjee asserts that one would modify the YTR application to allow a user to select and control a particular Leanback Screen in a session with other Leanback Screens (*see* Bhatta. Decl., ¶170), he has not explained how a POSITA would modify the system architecture such that, after multiple Leanback Screens have already been paired together in a session by the Lounge Server, the YTR application and/or Lounge Server would be configured to only control a particular Leanback Screen.

166. Further, a POSITA would not have been motivated to modify the YTR app to allow for a selection of a particular Leanback Screen because using multiple Leanback Screens was not even a prominent feature for the YTR app. Indeed, Google's own declarant, Janos Levai, testified that: (1) using YTR with multiple screens "sounds like an [edge] case," (2) "I don't remember using multiple screens at the same time," and (3) "why would I want to see a video on multiple screens?" Levai Dep Tr. at 110:23-111:16.

167. As another example, I disagree with Dr. Bhattacharjee's conclusory assertions that modifying the November 9, 2010 version of the YTR application to include the features of limitations 13.2 and 13.4 would have provided an obvious "benefit" that was "straightforward" to implement in the YTR application. Bhatta. Decl., ¶¶165, 174. First, this assertion ignores the system architecture of the YTR application, which, as explained herein, both teaches away from Google's proposed modifications and also renders such modifications more complicated than Google posits. *See* GOOG-SONOSNDCA-00075593, 94-95 (Google acknowledging that modifying the YTR pairing architecture "requires a complete redesign" and "lots of changes"). Additionally, the fact that Google did ***not*** include such functionality in the YTR application that was released on November 9, 2010 and that it took Google ***more than a year*** – and ***after*** the '615 Patent's invention date – to modify the YTR application to allegedly allow a user to select a particular Leanback Screen for playback/control suggests that Dr. Bhattacharjee's proposed modification was not such an obvious "benefit" that was "straightforward" to implement. *See* Bhatta. Decl., ¶170 (asserting that the ability to select a particular Leanback Screen for playback was first introduced in December 1, 2011 source code and subsequently included in a January 2012 version of the YTR application).

168. As yet another example, besides simply asserting without any explanation that it would have been obvious to modify the YTR application to "identify playback devices … on the local area network" because this was well known (*see* Bhatta. Decl., ¶¶165-66, 173), none of Dr. Bhattacharjee's obviousness arguments address specifically how the YTR application would have been modified to identify a Leanback Screen as one that is connected to ***the same local area network*** that the mobile phone running the YTR application is connected to, which is required by claim limitation 13.2, and a prerequisite to limitation 13.4(ii)'s requirement of an input that comprises "*a selection of the particular playback device from the identified playback devices connected to* ***the local area network***." For instance, Dr. Bhattacharjee does not even allege, much less analyze, what type of discovery mechanism the YTR application would use to identify Leanback Screens connected to the same local area network as the mobile phone running the YTR application or if such discovery mechanism would be a supplement to or a replacement of the

manual pairing mechanism used by the YTR application that relied on the intermediary Lounge Server and both devices being logged into the same YouTube account. Similarly, Dr. Bhattacharjee does not explain what type of discovery mechanisms that enabled a control device to identify playback devices connected to the same local area network as the control device were allegedly well-known at the time of Sonos's invention of the '998 Patent. Thus, Dr. Bhattacharjee's obviousness arguments fail for these reasons as well.

169. Finally, Dr. Bhattacharjee's theory for modifying the YTR application based on the "general knowledge" of a POSITA appears to rely on an assertion that "the YouTube Remote system already ***maintains unique identifiers*** for each Leanback Screen that has been paired with the YTR application in a session on the Lounge Server." *See* Bhatta. Decl., ¶174. However, Dr. Bhattacharjee fails to cite a single document to support this assertion.[32] Thus, for this reason alone, Dr. Bhattacharjee's obviousness allegation based on the general knowledge of a POSITA fails.

2. <u>The YTR Application + '998 Patent</u>

170. Dr. Bhattacharjee's apparent assertion that limitations 13.2 and 13.4 are obvious over the combination of the YTR application and the '998 Patent fails for many of the same reasons that his theory based on the YTR application and the "general knowledge" of a POSITA fails. Thus, my opinions and analysis above apply here. In addition, I also disagree with various of the specific arguments and characterizations that Dr. Bhattacharjee made with respect to the '998 Patent.

171. For instance, contrary to Dr. Bhattacharjee's suggestion, the '998 Patent does *not* disclose "***<u>transferring</u> playback***" of media from a control device to a playback device. Instead, the '998 Patent is directed to a "remote control" for ***<u>controlling</u> playback*** of media content on a "controlled device," "such as stopping playback of media content playing on the controlled devices or changing the media content playing on the controlled devices." *See, e.g.*, '998 Patent, 1:39-46; 3:34-55, Fig. 1. Thus, a POSITA would not have been motivated to seek out the '998 Patent for

[32] To the extent Dr. Bhattacharjee is relying on disclosure in the '998 Patent of "unique identifiers," he has not established that such identifiers were actually used or maintained in the alleged YTR application prior art system.

teachings related to the requirements of limitation 13.4, which are specifically directed to "*inputs to **transfer** **playback** from the control device to a particular playback device*." The '998 Patent does not include such teachings.

172. Take for example the primary disclosure in the '998 Patent that Dr. Bhattacharjee relies upon:

> In some examples, the user may also utilize the remote control application of remote control 75 to select one or more previously paired controlled devices, and to send control messages to one or more paired controlled devices. For example, the user may interact with user interface 84 and/or display 88 to interact with and control any available controlled devices.

'998 Patent, 10:63-11:6; *see also* Bhatta. Decl., ¶169. Nothing in this passage mentions or even suggests ***transferring*** playback from a "remote control" to a "controlled device." Accordingly, I disagree with Dr. Bhattacharjee that this passage teaches a selection of a particular "controlled device" to ***transfer*** playback to.

173. Moreover, in my opinion, this passage is ambiguous. For example, setting aside the fact that this disclosure does not teach transferring playback from a remote control to a controlled device, it is not clear to me that this disclosure in the '998 Patent teaches the ability of a "remote control" to present for selection multiple "controlled devices" that have already been paired in a session in a manner that would allow the user to select a particular "controlled device" for playback, as asserted by Dr. Bhattacharjee. *See* Bhatta. Decl., ¶169. Instead, consistent with my understanding of how the alleged prior art YTR application actually operated (*supra* ¶¶125-127), it appears that this passage may be referring to the ability to control one "controlled device," if that is the only "previously paired" "controlled device," or the ability to control all "controlled devices" collectively, if multiple "controlled devices" have been "previously paired" with a "remote control" in a session. Notably, Ramona Bobohalma, one of the named inventors of the '998 Patent and current employee of Google, stated that she couldn't explain what the "plain language" of this passage means. *See* Bobohalma Rough Dep. Tr., 104:21-105:3 (after testifying about what she allegedly was "thinking" about at the time, when pressed to explain what the "plain language" of the patent actually disclosed, she could not do so).

174. Further, in the context of the '998 Patent, and based on my understanding of the

operation of the alleged prior art YTR application, to the extent this disclosure in the '998 Patent does refer to the ability to select and control a specific one of multiple previously paired "controlled devices" (which is not clear as explained above), it would seem to me that this disclosure would be referring to the ability to use the "remote control application" to select and control one or more of the "controlled devices" that were "***previously*** paired" in a session and ***already all playing*** the same media. *See* Bhatta. Decl., ¶¶140, 144, 158, 160-162; GOOG-SONOS-WDTX-INV-00015423, 24. In other words, a user could, for example, select one of the "controlled devices" that was ***already playing*** media and then control the volume of that device. However, this has nothing to do with selecting a particular "controlled device" ***to transfer media playback*** from the "remote control" to the particular "controlled device," as required by limitation 13.4.

175. As another exemplary failure of Google's obviousness argument, the '998 Patent does not disclose a "*control device*" "*identifying playback devices connected to the local area network*," as required by limitation 13.2, and therefore, cannot make up for the lack of this functionality in the YTR application. While Dr. Bhattacharjee points to the '998 Patent's disclosure of "unique identifiers" for the "remote control" and "controlled devices" (*see* Bhatta. Decl., ¶168), Dr. Bhattacharjee acknowledges that the "unique identifiers" are maintained and used by the ***intermediary cloud server*** (sometimes referred to in the '998 Patent as the "intermediary" "network server" for providing a "network service" or "cloud service"). *Id.* The '998 Patent does *not* disclose the "***remote control***" receiving the "unique identifiers" of the paired "controlled devices" or otherwise identifying the "controlled devices" based on the "unique identifiers," let alone identifying the "controlled devices" as "controlled devices" connected to the same local area network as the "remote control." Dr. Bhattacharjee appears to agree with this; however, he nevertheless asserts that it would have been obvious to modify the alleged prior art YTR application to receive the "unique identifiers" described in the '998 Patent and thereby, identify the Leanback Screens as being connected to the same local area network as the mobile phone running the YTR application. *See* Bhatta. Decl., ¶¶165, 171-74. I disagree.

176. The '998 Patent teaches away from modifying the alleged prior art YTR application and/or '998 Patent in a manner that provides the "unique identifiers" of the "controlled

devices"/Leanback Screens to the "remote control"/YTR application and enables the "remote control"/YTR application to identify the "controlled devices"/Leanback Screens as being connected to the same local area network as the "remote control"/YTR application. For instance, the '998 Patent touts the system's reliance on the intermediary cloud-based "network service," including the maintenance and use of the "unique identifiers" by the "network service," as enabling the "remote control" and "controlled device[s]" to be paired in a session and "***not need[ing] to be connected to the same local area network***, nor in physical proximity to each other." *See* '998 Patent, 4:51-55 ("[B]y using the network service as an intermediary, the remote control and the controlled device, in various instances, may ***not need to be connected to the same local area network***, nor in physical proximity to each other.").

3. Secondary Considerations of Non-Obviousness

177. Assuming that I am correct that Google's Cast technology found in the YouTube apps and/or GPM app infringes '615 claim 13, it is my opinion that there exists secondary considerations evidence demonstrating that a POSITA would not have found '615 claim 13 obvious. For instance, I have seen evidence of Google receiving praise for its Cast technology that enables a Sender to transfer media playback responsibility from itself to a Receiver, which supports my opinion that '615 claim 13 would not have been obvious to a POSITA at the time of the invention.

178. As one example, a January 5, 2015 email from Google's employee highlights various quotes from prominent news sources touting Google's Cast technology for its "ease of use" and "heavy lifting from the cloud," among other reasons. *See* GOOG-SONOSNDCA-00073988, 88-89. I have reproduced some of these quotes below:

- "'I expect fans of Chromecast to appreciate the fact that ***Google is now casting a wider net***.' - USA Today"
- "'The tech giant wants to do for music streaming what it did for movie streaming -- ***make it simple to access entertainment from the Internet*** with your mobile device and play it on your living room gear.' – CNET"
- "'The audio device itself handles the streaming, so ***you don't have to leave your mobile gear turned on while you enjoy an hours-long playlist.***' – Engadget"

- "'Getting your audio to stream will work the same way it does for video - just hit the Cast button - but for audio, ***the stream will come directly from the cloud for improved sound quality.***' – Verge"

- "'While it ***sounds similar to what Apple has attempted to do with AirPlay*** speakers (which never really caught on), ***there's a key difference***: Instead of using your phone, tablet or computer as the source of the music or video, the ***Google Cast device pulls it down from the cloud. You get the best available quality and can still use your hand held device for other apps.***' - Wall Street Journal"

- "'Bluetooth speakers have taken off because they're a relatively simple solution for listening to music out loud-no cable required-but their simplicity can also be limiting for bigger sound systems. ***Google Cast for audio aims to provide listeners with a higher-fidelity option for connecting their mobile devices to new Internet-enabled speakers***.' – Fast Company"

- "'This is but a small piece of the overall ecosystem puzzle the company is trying to put together. With Chromecast becoming popular very quickly, ***Google is now looking to bring Cast into even more devices around the home ... At first glance it seems the company has taken the right first software and hardware steps to give it a strong start***.' – VentureBeat"

Id. The positive news coverage was well-received by Google's employees, including current CEO Sundar Pichai. *See id.*, 88 ("So great to see, well done all!").

179. Accordingly, it is my opinion that there exists secondary considerations evidence demonstrating that a POSITA would not have found '615 claim 13 obvious.

I declare under penalty of perjury under the laws of the United States that the foregoing is true and correct.

Dated: May 5, 2022

By: _____
Douglas C. Schmidt

EXHIBIT 1 – MATERIALS CONSIDERED BY DOUGLAS C. SCHMIDT

- U.S. Patent No. 9,967,615 ("the '615 Patent") and its prosecution history
- Google's Motion for Summary Judgment Pursuant to the Court's Patent Showdown Procedure and exhibits thereto, including the declarations of Dr. Bhattacharjee, Ms. Bobohalma, and Mr. Levai
- Sonos's Supp. Infringement Contention Chart for the '615 Patent, dated March 18, 2022
- Google's Invalidity Contentions w/r/t the '615 Patent, dated December 6, 2021
- Claim Construction materials, including Sonos, Inc.'s Opening Claim Construction Brief and exhibits thereto (Dkt. Nos. 184-185), such as the Expert Report of Dr. Schmidt on Claim Construction (Dkt. No. 185-8), Google LLC's Responsive Claim Construction Brief and exhibits thereto (Dkt. No. 200), Sonos, Inc.'s Reply Claim Construction Brief and exhibits thereto (Dkt. No. 202),
- Google's source code made available for inspection, including printouts of certain portions
- Final Deposition Transcripts of Vincent Mo, Umesh Patil, David Nicholson, and Janos Levai, and Rough Deposition Transcript of Ramona Bobohalma
- Examples of Accused Google Players and Google Pixel devices
- Sonos's Response to Google's Interrogatory No. 3, including Attachment A (dated Feb. 4, 2022)
- Google's Third Supp Objs & Resps to Sonos's Interrogatory Nos. 14-15 (dated Feb. 4, 2022)
- Google's Fifth Supp. Objs & Resps to Sonos's Interrogatory No. 12 (dated Apr. 7, 2022)
- Internal and publicly-available documents, including the following:

GOOG-SONOSWDTX-00006780	GOOG-SONOSNDCA-00073394
GOOG-SONOSWDTX-00006865	GOOG-SONOSNDCA-00071320
GOOG-SONOSWDTX-00006873	GOOG-SONOSNDCA-00056724
GOOG-SONOSWDTX-00039480	GOOG-SONOSNDCA-00071320
GOOG-SONOSWDTX-00039491	GOOG-SONOSNDCA-00072008
GOOG-SONOSWDTX-00039785	GOOG-SONOSNDCA-00071319
GOOG-SONOSWDTX-00039798	GOOG-SONOSNDCA-00071317
GOOG-SONOSWDTX-00039916	GOOG-SONOSNDCA-00071318
GOOG-SONOSWDTX-00041837	SONOS-SVG2-00067554
GOOG-SONOSWDTX-00043467	GOOG-SONOS-WDTX-INV-00015413
GOOG-SONOSWDTX-00051490	GOOG-SONOS-WDTX-INV-00015101
GOOG-SONOSWDTX-00052121	GOOG-SONOS-WDTX-INV-00015102
GOOG-SONOSWDTX-00053053	GOOG-SONOS-WDTX-INV-00015423
GOOG-SONOSNDCA-00073352	GOOG-SONOS-WDTX-INV-00001358
GOOG-SONOSNDCA-00073988	https://www.youtube.com/watch?v=fGMdg13YWB8
GOOG-SONOSNDCA-00075593	